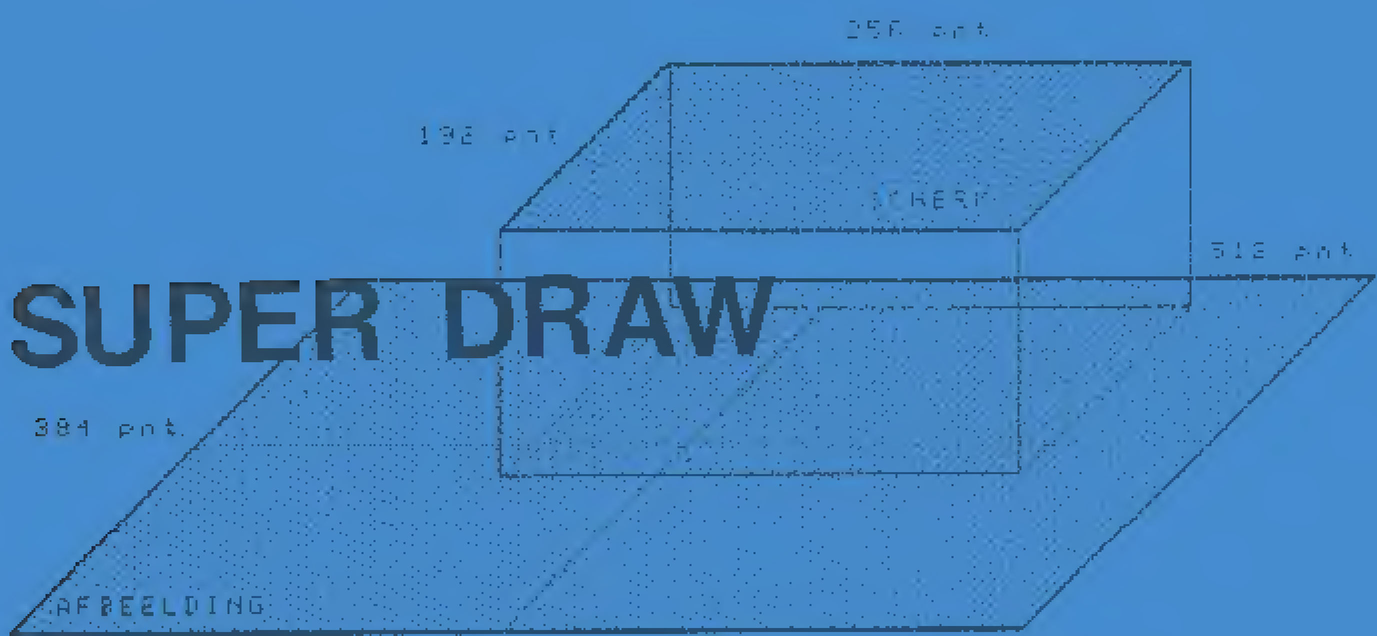
jrg: 3
nr: 2

mei 1984

ACORN Nieuws

figuur [illegible]
principe van windowing



SUPER DRAW

ACORN NIEUWS Uitg.Fed.Acorn Computerclubs Ned/B.
Verschijnt 6-8 x p/jr.

MET NADRUK EN DAAROM MET HOOFDLETTERS HET VOLGENDE:

COPIJ WELKE BREDER IS DAN 17 CM. GAAT ONHERROEPELIJK DE PRULLEBAK IN. EEN ENKELING HOUDT ZICH HIERAAN NIET, GELUKKIG DE MEESTEN WEL.

Ook bij deze Acorn Nieuws behoort een schijf welke naar alle regio's wordt gestuurd, mits de vorige retour is ontvangen. Tot op heden (17-4-83) is van 5 Rgio's de schijf nog niet retour.

De regio delft heeft nog enige kleuren kaarten in voorraad voor bestellingen TEL 070 - 899294

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100



HOPELIJK ME NIET VERTELD

Voorzitter: Th. van Kempen.
Het Puyven 71.
5672 RB Nuenen.
Telf 040 - 836210

Secretaris: A. Jongeling.
Klokkengietershoeve 3
7326 SB Apeldoorn.
Telf 055 - 417314

Penningmeester: G. Visser.
Haringvliet 391.
8032 HL Zwolle.
Telf 038 - 546561.

Giro 5244293
Bank 93.32.87.283
Beide t.n.v. Penningmeester Acorn computerclub.
Zwolle.

Redactie Acorn Nieuws.
H. Reinders.
Leeuwarderstraat 8
9718 HX Groningen.
Telf 050 - 125458

Bandjesarchief: H. Blijleven.
Noordzeestraat 47
1784 BL Den Helder
Telf 02230 - 16734

Datasheets: G. Akkermans.
Wikke 1
1273 BR HUIZEN.
Telf 02152 - 60294.

Drukwerkarchief: F. Monsato
Biesbosch 153
8032 VD Zwolle.

Uiterste datum inlevering copy:

NR 3. 27 - 4 - 1984.
NR 4. 25 - 5 - 1984.
NR 5. 27 - 7 - 1984.
NR 6. 24 - 8 - 1984.
NR 7. 28 - 9 - 1984.
NR 8. 26 -10 - 1984.
NR 1. 10 -12 - 1984.

Van bovenstaande data wordt niet afgeweken en plaatsing kan een nummer worden uitgesteld. 100 pagina's is het maximum, zo ook dit nummer.

Archiefdiensten.
Voor bandjes, datasheets, EPROM's en alle andere vormen van dienstverlening kunt u terecht bij de regionale archiefdiensten. Mocht in uw regio iets ontbreken, dan kan het regionaal bestuur u verder op weg helpen.
De landelijke archiefdiensten geven regelmatig een overzicht uit van wat er in die archieven aanwezig is. De regionale archiefdiensten kunnen daaruit dan een keus maken van wat zij in het archief willen hebben.
Een overzicht van de besturen en de archiefdiensten vindt u elders in dit nummer.
De "clubwinkel".

Geheugenkaart: 16 kByte extra in de Atom	fl. 35,--
Schakelkaart: meerdere EPROM's op Axxx (incl. 74LS133)	fl. 45,--
Programmerkaart: zelf programmeren van EPROM's	fl. 17,--
Herdruk Acorn Nieuws 1982: 97 pag. wetenswaardigheden	fl. 6,--
Jaargang 1983: totaal ruim 450 pag. compleet leesvoer	fl. 30,--

BELANGRIJK BESTELLEN BELANGRIJK

* Het bestellen van bovenstaande artikelen gelieve u te doen via
* uw regionale penningmeester. Alles wordt namelijk duurder en we
* houden de prijzen graag gelijk. Dat is alleen mogelijk door
* zoveel mogelijk de portokosten te drukken dmv collectieve
* bestellingen.
* Mocht u na het uitkomen van dit nummer toch rechtstreeks willen
* bestellen, dient u bij de bestelling een bedrag van fl. 4,--
* (zegge vier gulden) op te tellen voor extra verzendkosten.
* Vermeerdert u het bestelbedrag niet met die fl. 4,--, dan zal
* uw bestelling naar uw regionale penningmeester gestuurd worden
* met zijn eerstvolgende zending.

Belangstelling gevraagd.
Echt veel belangstelling is er niet getoond uit den lande. Het enige onderwerp wat wel veel gevraagd werd was de Big Benny.
Deze klok zal binnenkort dan ook in het assortiment van de clubwinkel opgenomen worden. Het is nog even wachten geblazen, want er moet eerst nog even gekeken worden in hoeverre de BB en andere ontwerpen elkaar bijten op poort PL 8.
Voor de reeds bestaande printplaten moet u ook nog enig geduld betrachten want de voorraad is op en het bijbestellen gaat niet zo snel als verwacht was. U kunt gewoon uw bestelling plaatsen (bij voorkeur bij uw reg. pennm. weet u nog) zodat de printen verstuurd worden zodra ze weer voorradig zijn.

Overzicht van de regionale bijeenkomsten.

3 MEI 1984 CENTRUM DE LANTAERN
UTRECHTSESTRAATWEG 4
20.00 UUR NIEUWEGEIN.

3 MEI 1984 BRABANT GEBOUW B.R.A.N.
HEVEL 7.
20.30 VORSTENBOSS.

4 MEI 1984 LIMBURG ZAAL HUIVENEERS
STATIONSSTRAT 35
19.30 UUR SITTARD

7 MEI 1984 ARNHEM t.o. POSTGIRO.
VELPERWEG 56-A (kelder)
20.00 UUR ARNHEM

9 MEI 1984 TWENTE ALMELO
Lokatie is gewijzigd. inlichtingen bij:
Ronald Boers.
Telf: 053 - 892335.

10 MEI 1984 NOORD MUNTSTERHOES
MUNTERHEERD
19.30 UUR GRONINGEN.

12 MEI 1984 NOORD-H DE KOPEREN KNOOP
LIMBURG VAN STIRUMSTRAAT
12.30 UUR AMSTERDAM

15 MEI 1984 BRABANT DE WERFF
V/D WERFFSTRAAT
20.00 UUR EINDHOVEN.

19 MEI 1984 COMPUTERDAG '84.
LEYSDREAM-COMPLEX.
11.00 UUR ROOSENDAAL.

1 JUNI 1984 NOORD-H

1 JUNI 1984 LIMBURG

4 JUNI 1984 ARNHEM

7 JUNI 1984 BRABANT VORSTENBOSCH.

8 JUNI 1984 DEN HAAG EXODUSKERK
BERESTEINLAAN 263
20.00 UUR DEN HAAG.

9 JUNI 1984 BELGIE OXACO-CEPENBERGCENTER
BORSBEEKSE STEENWEG 45
10.30 UUR BOECHOUT (bij HOVE).

R E G I O	voorzitter	secretaris	penn.meester/leden
noord	F. de Vries Scheldestraat 61 9406 PG Assen 05920-50880	K. Ottes Wilkemaheerd 67 9736 BN Groningen 050-417042	J. de Goede Hunze 26 9406 EE Assen 05920-50675 P.4411898
ov./geld.	J. Ballijns Antaresstraat 12 8303 AJ Emmeloord 05270-14410	G.J. Noorland Pr. Ireneweg 4 7433 DE Schalkhaar 05700-25294	G.J. Noorland Pr. Ireneweg 4 7433 DE Schalkhaar 05700-25294 B.146208234 ACC
twente	M. van Alphen Korianderhof 40 7641 XK Wierden 05496-2264	J. Laarman Potgieterstraat 13A 7422 XP Nijverdal 05486-17178	*W. Verhoeven *Witbreuksweg 385-306 *7522 ZA Enschede *053-893674 *P.4020195
n. holland	M. de Haan v. Hallstraat 537 1051 AC Amsterdam 020-827358	N. Stad Plataanweg 47 1544 PB Zaandijk 075-280808	N. Stad Plataanweg 47 1544 PB Zaandijk 075-280808 B.949672688 ACC
den haag	A. Slootweg Jaarsveldstraat 36 2546 CR den Haag 070-664422	H. vd Heyden Pr. Annalaan 574 2263 XZ Leidschendam 070-276726	R. Tiel Vredeoord 96 2544 TZ den Haag 070-294170 P.5469046 ACC
delft	A. Bliek Meerkoetlaan 93 2623 NG Delft 015-611578	M. Schroder Vlamingstraat 42 2611 KX Delft 015-143772	P. Kramer Oudraedweg 36 2612 SN Delft 015-141368 P.5467960 ACC
rotterdam	M. Oosterom Hooidrift 114A 3023 KV Rotterdam 010-255821	J. v Nieuwenhuyzen Brijhoekstraat 32 3223 BE H'voetsluis 01883-15129	R. de Haan Brasem 125 2986 HA Ridderkerk 01804-25160 B.442401094 ACC
centrum	F. Appelman Kastanjelaan 2 4131 AV Vianen 03473-76881	R.F. van Eijck Goudvink 21 3435 RH Nieuwegein 03402-46447	P. van Mourik Ruiterstede 60 3431 XN Nieuwegein 03402-48781 B.300024622 ACC

R E G I O	voorzitter	secretaris	penn.meester/leden
arnhem	R. van Dort Eliasstraat 27 6603 DA Alverna	H. Strumphler Kortenaerlaan 4 6881 ET Velp	J. Hartog Keyenbergseweg 60 6871 WK Renkum
zeeland	C. van Driel Walcherseweg 220 4334 NB Middelburg 01180-37246	H. Heringa Bosseweide 32 4431 AE 's Gravenpolder 01103-2437	H. Heringa Bosseweide 32 . . . 01103-2437 P.3009914
brabant o.	M.L. Verschuren Hosingenhof 30 5625 NL Eindhoven 040-416092	Th. van Kempen het Puyven 71 5672 RB Nuenen 040-836210	P. Ehrlich Roostenlaan 266 5644 BS Eindhoven 040-114183 B.528392794
limburg	H. Princen Europalaan w. 24 6075 AW Herkenbosch 04752-2534	E. Sanders Korvererf 1 6043 SL Roermond 04750-26051	W. Ernst Cypresstraat 94 6101 JX Echt 04754-5653 P.5457752
belgie	J. Lernout Parklaan 4 2540 Hove 03-4553473	J. Myngheer Fr.Verdonckstr. 39-8 1140 Brussel 02-2160404	R. Leyssens Oude Baan 127 3550 Heusden P.954723630149 ACC

bandjesarchief	datasheets	drukwerkarchief
H. Blijleven Noordzeestraat 47 1784 BL den Helder 02230-16734	G. Akkermans Wikke 1 1273 BR Huizen 02152-60294	F. Monsanto Biesbosch 153 8032 VD Zwolle

** De penningmeester van Twente bevindt zich tijdelijk voor stagewerkzaamheden in de VS.
Zijn taak wordt tijdelijk (waarschijnlijk tot augustus) overgenomen door:
C. Collet
prins Bernhardlaan 8
7622 BG Borne
074-661989
postgiro:4418159

DE CURSOR nr 2 (brabant)
De kleurenkaart herzien. Programma telmachine. Interpolator. joystickaanpassingen. basic-tekst-printer. Calculator.

ACORNTJESBROOD april (twente)
Graphic compress. BBC-basic op de ATOM. Geld verdeel spelletje. PL8 Connector. Magisch vierkant. Bespreking P=CHARME. Doolhof generator. Spreekwooden raden. 80 koloms VDU. Peano kromme.

DE STACKER maart (noord)
Kraken van Missile base. Hoe saved de Atom op een casette. De DOs controler. Casette catalog. Oplossing van een Disk drive probleem.

DE STACKER april (noord)
BBC-basic op de ATOM. Sneeuwloos plotten. Twee blokken tegelijk saven.

ROM januari (rotterdam)
turtle graphics. handleiding schakelsysteem. De DOS-BOX. over het samenstellen van een eigen box.

ROM april (rotteram)
De DOS en de 2e 16k kaart. divere wetenswaardigheden.

DATA-CHECK nr 1 (den haag)
Locig probe. vervolg vertaling hand-leiding. Diverse kleine programmaatjes. De ASBK box.

DATA-CHECK nr 2 (den haag)
Extra scherm. Disk-drive qulifier problemen. diverse kleine programma's.

Centerfold (centrum)
Software buffer.

Bronsgroen Eikeltje nr2. (limburg)
microsoft-basic. bank-switsing met 64K-kaart. Snake. Fast-plot. Kwardraten van Mezirac.

Bronsgroen Eikeltje nr3 (limbutg)
uiteenzetting van de COS. Eprom programmer programma. Alles over READ DATE en RESTORE. Vervolg op INFOMASTER.

ATOMIX nr 7 (delft)
High resolution plotter.

ATOMIX nr 8 (delft)
Zelfportret. Grafics source code. Grafics. Programmeren in FORTH. De Atom als terminal. On goto/ gosub source code.

ATOMIX nr 9 (delft)
Tips voor 64k kaart bezitters. verborgen intruktie's van de 6502. Circle en Delete. Een schakelkaart operating system.

Literatuuroverzicht van verschenen boeken rond de Acorn Atom:

1. Getting aquainted with your Acorn Atom.
   Een boek gevuld met allerlei mogelijke spelletjes, alsmede een handleiding voor zelf spelletjes te maken.

2. Wake up your atom.
   Ook weer een boek met allerlei spelletjes, w.o. tennis. De meeste spellen zijn gevuld met colour-statements, doch draaien net zo goed op een zwart-wit t.v. of monitor.

3. Practical programs for the Acorn Atom and the BBC computer.
   Hierin een groot aantal programma's voor beide acorn computers, en verspreid over een aantal hoofdstukken, o.a. graphies, spelltejes, woordmánipulaties, rekenprogramma's en een compiler.

4. Atom business.
   Een boek met programma's toegespitst op de atom voor zakelijk gebruik. O.a.: een programma om labels met naam adres, woonplaats uit te printen; een optel-programma; een heus grootboekprogramma, waarmee je uitstekend de financiën van de vereniging mee kunt berekenen en een wachttijd-simulatieprogramma. Tevens nog een programma om het voorraadbestand op bij te houden.

5. The Atom magic book.
   Bij velen bekend, neem ik aan.

6. Splitting the Atom.
   Een boek met overzichten van atom adressen in rom.
   Te bestellen bij: Focusplan,
   57 Westgate,
   Cleckheaton,
   W.-Yorks BD - 195 14.

7. Het laatst bekende boek:
   Atomic Machine code.
   Een boek wat 23 volledig uitgelegde machine code programa's bevat. O.a. data-sorts, mode 4 characters, games, pools prediction, tool-kit.
   Het boek kost 5.75
   Boek en cassette (source code) kost 15.50
   Book en cassette (ready to run) kost 15.50
   Cassette kost 11.50 (alle prijzen engels ponden)
   Te bestellen bij:
   ECCE Productions,
   3/73 Station Road,
   Sidcup, Kent, DA15 7DR.

Zijn er nog meer bekende boeken?
Laat het iedereen weten, d.m.v. Acorn nieuws.

Het programma INFOMASTER ligt in het geheugen-gebied van #8200 tot #A000. Dit is 7.5 Kb geheugen en er moet dus in het hoge gebied gestapeld zijn. Door deze keus kan er zoveel mogelijk data opgeslagen worden in de computer. De arrays liggen vanaf #2800. De data wordt opgeslagen in het geheugen gebied beginnende met #2902. U kunt dus vullen tot #3C00 of #4000 of zelfs tot #8000 voor de bezitters van bijvoorbeeld een CMOS-kaart.

Het programma wordt geladen met *LOAD"INFOMASTER" 8200. Het programma runt vanuit textspace 84 . Van #8220 tot #8400 ligt assemblercode.

OPMERKING

Het programma is ten alle tijde te stoppen via de escapetoets. Hierna kan het programma in textspace 82 of 84 weer worden gestart zonder dat de DATABASE beschadigd is. Zelfs na een break blijft de DATABASE onbeschadigd.

2.0 TOEPASBAARHEID

Dit programma is een DATABASE-programma. Hiermee is het dus mogelijk met records, die uit een of meerdere items bestaan, te werken. Het aantal records is maximaal 65535 en de beperkingen van het geheugen zullen dus eerder limiterend werken.

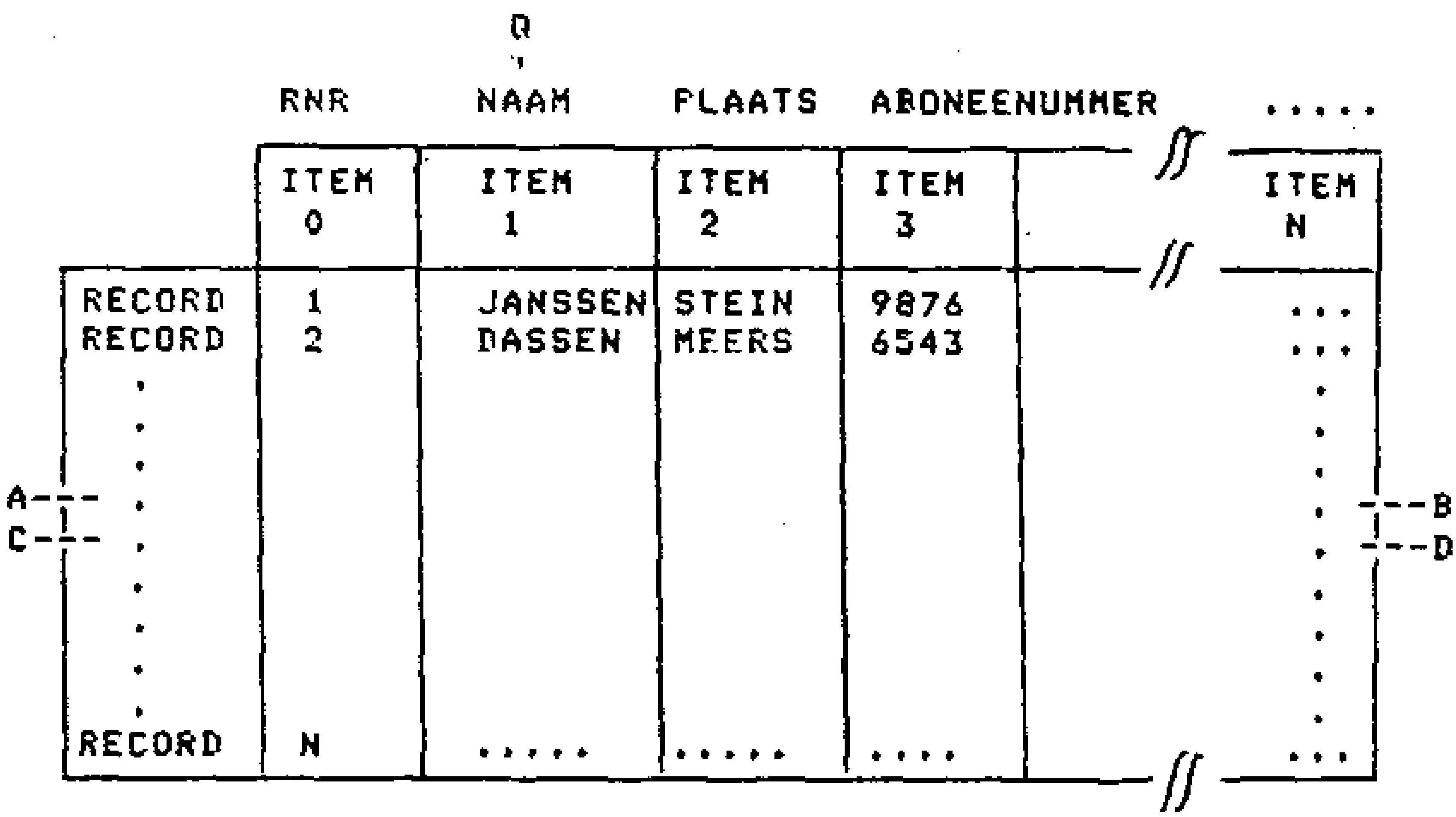
Een item neemt in een record 1 , 2 , of 4 bytes in beslag. Naar een ASCII-string wordt d.m.v. een 2 bytes adres in een ASCII-file verwezen. Dit ASCII-file bevat iedere string slechts eenmaal. Daardoor wordt er optimaal gebruik gemaakt van het geheugen. Het aantal items is maximaal 255. Dit kan bereikt worden bij 1 byte opslag. Het gemiddelde ligt ongeveer op 2 bytes opslag , zodat er dan nog zelfs met 128 items gewerkt kan worden. Een telefoonbestand kan bijvoorbeeld uit 7 items bestaan. Ten behoeve van stamboomonderzoek heb ik een databestand aangemaakt, waarbij per persoon (een record) zelfs 44 items zijn gedefinieerd.

In het programma is ook een mogelijkheid tot MERGEN ingebouwd, waardoor zelfs zeer grote bestanden kunnen worden verwerkt. De bestanden mogen zelfs groter zijn dan het actuele computergeheugen. Zo heb ik een in drieen gedeeld bestand, waarin alle huwelijken te STEIN voor 1800 (962 stuks!!) zijn opgenomen. Ieder huwelijk bestaat uit 8 items.

Ook is het mogelijk om met hexadecimale getallen te werken, zonder dat het #-teken te hoeven in te typen. Dit kan bijvoorbeeld handig zijn als U een archief voor cassette-bandjes wilt opzetten, waarin U ook begin- ,eind- en startadres wilt opnemen. Deze hexadecimale getallen worden in 2 bytes opgeslagen.

3.0 DATABASE

Een hoeveelheid gegevens kunnen in het algemeen in een bestand ondergebracht worden. Dit noemen we een DATABASE. Een DATABASE is op te vatten als een in rijen en kolommen ingedeeld groot vel papier. De rijen worden ook records genoemd; de kolommen items. De records worden genummerd vanaf 1. Dit gaat bij INFOMASTER automatisch in item 0 met de symbolische naam RNR (Recordnummer) . De symbolische namen van de andere items moeten door U opgegeven worden. Een tekening maakt dit misschien wat duidelijker.



4.0 MENU-BESCHRIJVING

Als U het programma INFOMASTER runt dan krijgt U het hoofdmenu te zien.

Dit menu geeft de volgende keuze mogelijkheden :

1. = EERSTE OPSTART

2. = LOAD DATA BESTAND

3. = SAVE DATA BESTAND

4. = UPDATEN

5. = PRINTEN

6. = SORTEREN

7. = MERGEN

8. = OVERZICHT

Uit het bovenstaande menu kunt U een keus maken. Natuurlijk is het zo dat U, als er nog geen data aanwezig is, slechts de keus heeft uit de eerste twee mogelijkheden. Deze keuze mogelijkheden worden hieronder uitvoerig besproken.

4.1 EERSTE OPSTART

Deze mogelijkheid heeft U nodig bij het aanmaken van een nieuwe database. Het is erg zinvol als U van te voren even opschrijft hoe U databestand er ongeveer gaat uitzien. Dan kunt U op de eerste vraag AANTAL ITEMS? het juiste antwoord geven (N) . Met item0 (nummering) hoeft U geen rekening te houden.

Hierna wordt in een loop voor de items 1 tot en met N de symbolische naam gevraagd. Bijvoorbeeld :

1. NAAM

2. PLAATS

3. etc.

Nu moeten we gaan opgeven wat voor een soort inhoud in een item komt. We onderscheiden 2 hoofdmogelijkheden :

1. ALFANUMERIEK
   Als in een item cijfers en letters bewaard moeten worden (tekst) dan moeten we het item ALFANUMERIEK verklaren.

2. NUMERIEK
   Als in een item getallen bewaard moeten worden dan moet het item NUMERIEK verklaard worden. Om een zo nuttig mogelijk gebruik van het geheugen te kunnen maken moeten we nu gaan opgeven hoeveel bytes gereserveerd moeten worden om het getal op te slaan. Dit gebeurt na de vraag 1,2 of 4 bytes of hexadecimale getallen (-1). Hier zijn de vier mogelijkheden :

   1. EEN BYTE GETALLEN
      Indien het op te slaan getal ligt tussen 0 en 255, dan moet U opgeven 1. Dit is bijvoorbeeld zo met dagen,maanden etc.

   2. TWEE BYTES GETALLEN
      Als het op te slaan getal ligt tussen 0 en 65535 dan moet U hier 2 opgeven. Dit komt voor bij bijvoorbeeld jaartal of de cijfercombinatie van de postcode.

   3. VIER BYTES GETALLEN
      Indien U met grote of negatieve getallen wilt werken dan moet U hier 4 opgeven.Er wordt dan gewerkt met de normale integer getallen (grenzen ongeveer + en - 2 miljard) .

   4. HEXADECIMALE GETALLEN
      Wilt U werken met hexadecimale getallen dan moet U hier -1 opgeven.De getallen kunnen daarna zonder & ingetypt worden.

De EERSTE OPSTART maakt automatisch 1 lees record aan. Er wordt nu naar de KOPTEKST gevraagd. Dit is de tekst die later boven iedere pagina geprint kan worden. Het zelfde geldt voor de DATUM die U moet opgeven.

4.2 LOAD DATA BESTAND

Op de vraag NAAM? dient U de naam op te geven waaronder U de vorige keer het bestand op band heeft gezet. Na het laden wordt de DATUM gevraagd. Vul deze in zodat U na wijzigen en vervolgens op band zetten de administratie goed bijhoudt.

4.3 SAVE DATA BESTAND

Aan een databestand kunt U een naam toekennen. Met deze naam wordt uw bestand vervolgens op band gezet. Het begin- en eindadres van het geheugengebied dat op band wordt gezet kunt U vinden als U het overzicht opvraagt (zie 4.8).

4.4 UPDATEN

Met behulp van UPDATEN is het mogelijk om records toe te voegen, te verwijderen of te wijzigen.
We onderscheiden twee hoofdmodes : RECORDMODE en ITEMMODE

1. RECORDMODE

Als we, vanuit het HOOFDMENU, UPDATEN kiezen komen we binnen in de RECORDMODE. Op de vraag RECNR? zijn de volgende antwoorden mogelijk:

een (legaal) getal
Ieder record in het databestand heeft een nummer. Als U dus hier een legaal nummer intypt komt het record met dit nummer aan de beurt. Het programma gaat dan over in de ITEMMODE en begint bij ITEM1.

H(ELP)
Het programma laat nu zeer in het kort zien welke commando's mogelijk zijn.

^ of een return
Terug naar HOOFDMENU.

NEW
Nu worden aan het bestand nieuwe records toegevoegd. We gaan nu direct over in de ITEMMODE. Uit deze mode is te komen met of ^ of DELREC.

RENUM
Ieder record heeft een eigen nummer. Ze worden genummerd in volgorde zoals ze met NEW zijn aangemaakt. Heeft U later uw bestand op een ander kenmerk gesorteerd (bijv. op naam) dan kunt U de records hernummeren (renumber). Later ziet U dan bij uitprinten van het RNR (recordnummer) dat U een oplopende teller heeft.

LOCx
Localiseer item met nummer x. Dit is een hulpmiddel van een string of getal zoals dat in het bestand voorkomt. Stel dat ITEM1 de NAAM bevat dan is na LOC1 de volgend vraag WELKE NAAM ?. U typt in JANSSEN en nu krygt U alle recordnummers te zien van alle JANSSEN's in uw bestand.

Lx
List de inhoud van alle items met nummer x op het scherm.

Qx
Quick mode voor item met nummer x. In deze mode wordt bij UPDATEN alleen dit item behandeld. U loopt dus door uw bestand volgens de lijn Q-Q. (zie tekening) Uit deze mode is te komen door terug te gaan naar het HOOFDMENU.

OPMERKING

LOCx en Lx zijn af te breken door op de shifttoets te drukken.

2. ITEMMODE
========

U doorloopt in de ITEMMODE het actuele record van ITEM1 tot ITEMN. Dit gebeurt in de numerieke volgorde (zie tekening lijn A-B). Vervolgens komt het volgende record (in het geheugen) aan de beurt. Dit gaat weer van ITEM1 t/m ITEMN volgens de lijn C-D etc. (zie tekening) Van het actuele record en het actuele item ziet U de inhoud. De computer staat op uw input te wachten. Legale antwoorden zijn :

een nieuwe waarde
De vorige inhoud van dit item wordt vervangen door deze nieuwe waarde.

een return
Ga naar het volgende item van dit record. Of als dit het laatste item is ga dan naar het eerste item van het volgende record. Er wordt niets verander aan de inhoud van het item.

-
Ga terug naar recordmode.

/
Ga direct naar het eerste item van het volgend record.

]
Geef aan dit item dezelfde waarde als het overeenkomstige item van het vorige record.

\
Wis van dit item de aanwezige inhoud uit. Als dit de laatst ingetypte string betreft, dan wordt deze ook uit het ASCII-file verwijderd.

?
Laat zien aan welk record we bezig zyn.

Qx
Ga snel naar item met nummer x van dit record.

DELREC
Verwijder het actuele record uit het databestand.

H(ELP)
Help mode. Laat de vorige commando's zien.

4.5 PRINTEN

Als U gekozen heeft voor PRINTEN dan volgt nu het PRINTMENU. Op een return of ^ gaat U terug naar het HOOFDMENU. Er zijn de volgende mogelijkheden in het PRINTMENU :

1. PRINT
   Het bestand wordt uitgeprint zoals U dat zelf gespecificeerd heeft (zie verderop). Dit uitprinten kan zowel naar het scherm maar ook naar een echte printer. Stuurt U het naar een printer dan volgt de vraag VANAF PAGNR ?. Dit is handig bij grote bestanden als U vanaf een bepaalde pagina geprint wilt hebben. Een return is voldoende als U alles geprint wilt hebben.

   OPMERKING

   Het printen is te onderbreken door even de shifttoets in te drukken totdat het record (regel) klaar is met printen.

2. WIJZIGEN KOP
   De volgende zaken die in U printout bovenaan de bladzijde kunnen verschijnen kunt U nu eventueel wijzigen.

   a. De symbolische itemnamen.

   b. De KOPTEKST

   c. De DATUM

3. PRINTER SETTING
De volgende zaken kunt U met J(A) of N(EE) beinvloeden. Een return of een ander karakter verandert niets.

PRINTER
Wel of niet naar een echte printer. Nee is alleen naar scherm.

KOP
Wel of niet een kop boven de te printen pagina's.

PAG NR
Wel of niet het paginanummer boven de te printen pagina's.

DATUM
Wel of niet een datum boven de te printen pagina's.

4. HOE ITEMS PRINTEN
Hier kunt U opgeven hoe U de items geprint wilt hebben.

ITEM
Hier geeft U op of U een item uberhaupt geprint wilt hebben. Zo ja verder met :

SUBKOP
Wilt U boven iedere kolom de symbolische itemnaam geprint hebben ?.

LINKS PRINTEN
Hier opgeven of in een kolom naar rechts of links opgeschoven moet worden bij printen.

NULLEN ONDERDRUKKEN
Dit geldt alleen voor numerieke items. Indien J(A) dan worden de nullen bij printen vervangen door spaties.

PRINTBREEDTE
Hier opgeven hoe breed U een kolom wilt hebben bij printen. Er gaat niets verloren als U de breedte te klein kiest. Het item schuift dan de volgende kolom in. Op de bovenstaande vragen kunt U antwoorden met :

a. een return

Naar het volgende onderdeel

b. ^

terug naar PRINTMENU.

c. Q:x

ik wil snel naar item x om daar op te geven hoe ik dat item geprint wil hebben. Niet gebruiken bij PRINTBREEDTE.

d. J(A) of N(EE)

5. OMKEER RESET
Als U de kolommen in dezelfde volgorde wilt hebben als de items in het archief staan moet U dit kiezen. Dit is alleen nodig als U de omkering bij printen wilt verwijderen.

6. OMKEREN BY PRINTEN
Als U van de normale volgorde van de kolommen afwilt komt U hier terecht. Nadat U gezien heeft in welke volgorde de items normaal liggen, krijgt U de kans om te wijzigen. U kunt ook een item in meerdere kolommen tegelijkertijd laten afdrukken. Op return komt steeds het volgende item. Na ^ komt U terug in het PRINTMENU.

7. SELEKTEREN
Het is mogelijk als U niet het hele bestand wilt geprint hebben om hier een selektie op te geven. Van een door U te kiezen item kunt U grenzen opgeven. Dit kan zowel een string als ook een getalselektie zijn. Bijv. U wilt alleen de namen van JANSSEN tot PIETERSEN eruit hebben. Of ook bijv. de huisnummers (=bijv. item3) die tussen 10 en 20 liggen. Heeft U het te kiezen item voor selektie negatief opgegeven (dus bijv. -3) dan volgt een NIET funktie over de selektie. Met andere woorden bijv. alle huisnummers < 10 en > 20 worden wel geprint.

8. GEEN SELEKTIE
Als U alle records weer mee wilt laten doen bij het printen dan komt U hier terecht.

OPMERKING

Al de gegevens die U in het printmenu opgeeft worden opgeslagen in het databestand. Hierdoor gaan ze bij saven mee naar de band, zodat ze bij een volgende gelegenheid weer gelijk ter beschikking staan.

4.6 SORTEREN

Nadat U te zien heeft gekregen welke items er zijn, volgt er op welk itemnummer het bestand gesorteerd staat. De allereerste keer is dit natuurlijk op RNR (item 0). Dit is namelijk de volgorde waarin de records zijn aangemaakt. Als het bestand op een numeriek item gesorteerd is dan volgt als dit van hoog naar laag gebeurd is de toevoeging HL. Geen toevoeging betekent dus van laag naar hoog gesorteerd.
Nu volgt de vraag SORT NR ? Op een return gaat U terug naar het HOOFDMENU en gebeurt er verder niets. Geeft U een itemnummer op dan volgt als het een nummeriek item is de vraag HOOG->LAAG J/N. Hiermee kunt U dus opgeven of U dalende of stijgende volgorde wilt hebben.
Alfanumerieke items worden altijd op ASCII-waarde gesorteerd dus van A-Z.
Door de keuze van het sorteeralgoritme (bubblesort) wordt de vorige sortering zo lang mogelijk gehandhaafd. Dus ondersorteringen zijn mogelijk. Heeft U bijvoorbeeld een bestand met namen en plaatsen en wilt U een lijst gesorteerd op naam maar van alle JANSSENS de plaatsnamen ook gesorteerd, dan sorteerd U eerst op plaats en daarna op naam.
Tijdens het sorteren verschijnen getallen op het scherm. Hieraan ziet U hoeveel records er nog gesorteerd moeten worden. Het programma heeft direct door als er niets meer te sorteren valt.

4.7 MERGEN

Met MERGEN krijgt U te maken als U met zeer grote bestanden gaat werken. Er zijn mogelijkheden om het bestand in gedeeltes te splitsen, of ook omgekeerd bestanden met elkaar te combineren.
Een bestand bestaat uit een RECORD-file en een ASCII-file. Als U gaat mergen is het nuttig om eerst het ASCII-file te saven. Daarna kunt U het RECORD-file of een gedeelte ervan saven.
Nadat het aantal records is getoond en gevraagd naar SAVE of LOAD wordt naar REC of ASCII-file gevraagd. Als U kiest voor REC file wordt vervolgd met VANAF RECNR ?. U geeft nu op vanaf welk recordnummer U naar de band wilt hebben. Dit gedeelte wordt daarna uit het bestand verwijderd en U kunt nu weer nieuwe records intypen. Het ASCII-file wordt nu weer mogelijk groter maar blijft toepasbaar voor de oudere records.
Het is ook mogelijk om een recordfile dat U eerder op band hebt gemerged bij een actueel recordfile bij te laden. Als U bedenkt dat het ASCII-file achter het recordfile ligt zal het duidelijk zijn dat dit overschreven kan worden. Het programma stopt dan met ASCIIFILE>. U start INFOMASTER weer en dan kiest U MERGE en U laadt het eerder gesaved ASCII-file. Dit wordt nu automatisch op de goede plaats geladen. En nu kunt U weer verder met het door U samengestelde bestand.

4.8 OVERZICHT

Bij het overzicht krijgt U het volgende te zien :

1. Symbolische itemnamen met itemsoort.

2. Omvang totaal databestand.

3. Omvang van het recordgedeelte hieruit.

4. Omvang van het ASCII-gedeelte hieruit.

5. Het aantal records dat in het bestand aanwezig is.

5.0 CASSETTE-ARCHIEF ALS VOORBEELD

Ter illustratie volgt hieronder een voorbeeld van een Data-Base gecreeerd met behulp van INFOMASTER.

```
>RUN
               INFOMASTER
1=EERSTE OPSTART       2=LOAD
3=SAVE                 4=UPDATEN
5=PRINTEN              6=SORTEREN
7=MERGEN               8=OVERZICHT
KEUS?1
AANTAL ITEMS?9

NAAM
ITEM 1?NAAM
ITEM 2?BAND
ITEM 3?TELLER
ITEM 4?START
ITEM 5?EIND
ITEM 6?EXEC
ITEM 7?SOORT
ITEM 8?CAS
ITEM 9?DATUM

NUMERIEK
OF ALFANUMERIEK (N/A)

ITEM 1 = NAAM (N/A) ?A
ITEM 2 = BAND (N/A) ?N

1,2,OF 4 BYTES -
OF 2BYTES HEXGETALLEN (-1)
(1/2/4/-1) ?2
ITEM 3 = TELLER (N/A) ?N
```

```
1,2,OF 4 BYTES -
OF 2BYTES HEXGETALLEN (-1)
(1/2/4/-1) ?2
ITEM 4 = START (N/A) ?N

1,2,OF 4 BYTES -
OF 2BYTES HEXGETALLEN (-1)
(1/2/4/-1) ?-1
ITEM 5 = EIND (N/A) ?N

1,2,OF 4 BYTES -
OF 2BYTES HEXGETALLEN (-1)
(1/2/4/-1) ?-1
ITEM 6 = EXEC (N/A) ?N

1,2,OF 4 BYTES -
OF 2BYTES HEXGETALLEN (-1)
(1/2/4/-1) ?-1
ITEM 7 = SOORT (N/A) ?A
ITEM 8 = CAS (N/A) ?A
ITEM 9 = DATUM (N/A) ?N

1,2,OF 4 BYTES -
OF 2BYTES HEXGETALLEN (-1)
(1/2/4/-1) ?4
```

```
KOPTEKST? CASSETTEARCHIEF

DATUM? 4-8-83

KEUS?



              INFOMASTER
1=EERSTE OPSTART        2=LOAD
3=SAVE                  4=UPDATEN
5=PRINTEN               6=SORTEREN
7=MERGEN                8=OVERZICHT
KEUS? 4

RECNR? 1

RECORD 1
1 NAAM    ? INFOMASTER
2 BAND  0? 202
3 TELLER  0? 60
4 START  0? 8200
5 EIND  0? 9FA5
6 EXEC  0? 84
7 SOORT   ? APP
8 CAS   ? V3
9 DATUM  0? 830614

RECNR? NEW

NIEUW RECORD 2
1 NAAM ? DATPROC
2 BAND ? J
3 TELLER ? 2
4 START ? 2800
5 EIND ? 3C00
6 EXEC ? 29
7 SOORT ? J
8 CAS ? J
9 DATUM ? 830610

NIEUW RECORD 3
1 NAAM ? SOURCEMAKER
2 BAND ? 201
3 TELLER ? 2
4 START ? 4000
5 EIND ? 509B
6 EXEC ? 42
7 SOORT ? SYSTEM
8 CAS ? J
9 DATUM ? 830613

NIEUW RECORD 4
1 NAAM ? DELREC

RECNR? ↑

KEUS? 8
```

```
ITEM0=RNR              INT*2
ITEM1=NAAM             ASCII
ITEM2=BAND             INT*2
ITEM3=TELLER           INT*2
ITEM4=START            HEX*2
ITEM5=EIND             HEX*2
ITEM6=EXEC             HEX*2
ITEM7=SOORT            ASCII
ITEM8=CAS              ASCII
ITEM9=DATUM            INT*4

DATABESTAND            2900 -2A86
RECORDS                29A2 -29E4
ASCIIFILE              2A00 -2A86
AANTALRECORDS     3

KEUS? 5

PRINTMENU
1=PRINT
2=WYZIGEN KOP
3=PRINTER SETTING
4=HOE ITEMS PRINTEN
5=OMKEER RESET
6=OMKEREN BY PRINTEN
7=SELEKTEREN
8=GEEN SELEKTIE

UW KEUS? 4

   0       RNR
ITEM=J ?
SUBKOP=J ?
LINKS PRINTEN=J ?
NULLEN ONDERDRUKKEN=N ?J
PRINTBREEDTE=    4 ?


   1       NAAM
ITEM=J ?
SUBKOP=J ?
LINKS PRINTEN=J ?
PRINTBREEDTE=    9 ?14

   2       BAND
ITEM=J ?
SUBKOP=J ?
LINKS PRINTEN=J ?N
NULLEN ONDERDRUKKEN=N ?J
PRINTBREEDTE=    9 ?5
```

```
   3       TELLER
ITEM=J ?
SUBKOP=J ?
LINKS PRINTEN=J ?N
NULLEN ONDERDRUKKEN=N ?J
PRINTBREEDTE=    9 ?7

   4       START
ITEM=J ?
SUBKOP=J ?
LINKS PRINTEN=J ?N
NULLEN ONDERDRUKKEN=N ?J
PRINTBREEDTE=    9 ?6

   5       EIND
ITEM=J ?
SUBKOP=J ?
LINKS PRINTEN=J ?N
NULLEN ONDERDRUKKEN=N ?J
PRINTBREEDTE=    9 ?6

   6       EXEC
ITEM=J ?
SUBKOP=J ?
LINKS PRINTEN=J ?N
NULLEN ONDERDRUKKEN=N ?J
PRINTBREEDTE=    9 ?6

   7       SOORT
ITEM=J ?
SUBKOP=J ?
LINKS PRINTEN=J ?N
PRINTBREEDTE=    9 ?7
```

```
   8       CAS
ITEM=J ?
SUBKOP=J ?
LINKS PRINTEN=J ?N
PRINTBREEDTE=    9 ?5

   9       DATUM
ITEM=J ?
SUBKOP=J ?
LINKS PRINTEN=J ?N
NULLEN ONDERDRUKKEN=N ?J
PRINTBREEDTE=    9 ?7

UW KEUS?H

PRINTMENU
1=PRINT
2=WYZIGEN KOP
3=PRINTER SETTING
4=HOE ITEMS PRINTEN
5=OMKEER RESET
6=OMKEREN BY PRINTEN
7=SELEKTEREN
8=GEEN SELEKTIE

UW KEUS?3
PRINTER=N ?J
KOP=N ?J
PAG NR=N ?J
DATUM=N ?J

UW KEUS?1
VANAF PAGNR?
```

4-8-83 PAG

CASSETTEARCHIEF
****

RNR	NAAM	BAND	TELLER	START	EIND	EXEC	SOORT	CAS	DATUM
1	INFOMASTER	202	60	8200	9FA5	84	APP	V3	830614
2	DATPROC	202	2	2800	3C00	29	APP	V3	830610
3	SOURCEMAKER	201	2	4000	5C9B	42	SYSTEM	V3	830613

```
UW KEUS?

KEUS?
           INFOMASTER
1=EERSTE OPSTART      2=LOAD
3=SAVE                4=UPDATEN
5=PRINTEN             6=SORTEREN
7=MERGEN              8=OVERZICHT
KEUS?6
```

```
PRINTMENU                          ITEM0=RNR          INT*2
1=PRINT                            ITEM1=NAAM         ASCII
2=WYZIGEN KOP                      ITEM2=BAND         INT*2
3=PRINTER SETTING                  ITEM3=TELLER       INT*2
4=HOE ITEMS PRINTEN                ITEM4=START        HEX*2
5=OMKEER RESET                     ITEM5=EIND         HEX*2
6=OMKEREN BY PRINTEN               ITEM6=EXEC         HEX*2
7=SELEKTEREN                       ITEM7=SOORT        ASCII
8=GEEN SELEKTIE                    ITEM8=CAS          ASCII
                                   ITEM9=DATUM        INT*4
UW KEUS?1                          GESORTEERD    0
VANAF PAGNR?                       SORT NR ?1
                                      3    2
                                   KEUS?5
4-8-83

                                    CASSETTEARCHIEF
                                    ===============

GESORTEERD OP NAAM

 RNR NAAM             BAND TELLER START  EIND  EXEC  SOORT   CAS  DATUM

 2   DATPROC           202      2  2800  3C00    29     APP   V3 830610
 1   INFOMASTER        202     60  8200  9FA5    84     APP   V3 830614
 3   SOURCEMAKER       201      2  4000  509B    42  SYSTEM   V3 830613

UW KEUS?H

PRINTMENU                          UW KEUS?7
1=PRINT
2=WYZIGEN KOP                      ITEM0=RNR          INT*2
3=PRINTER SETTING                  ITEM1=NAAM         ASCII
4=HOE ITEMS PRINTEN                ITEM2=BAND         INT*2
5=OMKEER RESET                     ITEM3=TELLER       INT*2
6=OMKEREN BY PRINTEN               ITEM4=START        HEX*2
7=SELEKTEREN                       ITEM5=EIND         HEX*2
8=GEEN SELEKTIE                    ITEM6=EXEC         HEX*2
                                   ITEM7=SOORT        ASCII
                                   ITEM8=CAS          ASCII
                                   ITEM9=DATUM        INT*4
                                   SELECTIE OP NR?7

UW KEUS?1                          VAN ?APP
VANAF PAGNR?
                                   TOT ?APP

4-8-83                                                                PAG

                                    CASSETTEARCHIEF
                                    ===============

GESORTEERD OP NAAM
SOORT VAN APP TOT APP

 RNR NAAM             BAND TELLER START  EIND  EXEC  SOORT   CAS  DATUM

 2   DATPROC           202      2  2800  3C00    29     APP   V3 830610
 1   INFOMASTER        202     60  8200  9FA5    84     APP   V3 830614
```

```
UW KEUS?

KEUS?4

RECNR?H

↑ NEW RENUM LOCX LX QX
OF 'N GETAL<=3
X=ITEMNR<=9
RECNR?2

RECORD 2
1 NAAM   DATPROC?
2 BAND  202?
3 TELLER   2?Q8
8 CAS   V3?Q3
3 TELLER   2?↑

RECNR?Q1

RECNR?2

RECORD 2
1 NAAM   DATPROC?

RECORD 1
1 NAAM   INFOMASTER?

RECORD 3
1 NAAM   SOURCEMAKER?

RECNR?
```

```
KEUS?4

RECNR?1

RECORD 1
1 NAAM   INFOMASTER?H

↑ / 1 \ QX DELREC
1 NAAM   INFOMASTER?/

RECORD 3
1 NAAM   SOURCEMAKER?↑

RECNR?RENUM

KEUS? 5

PRINTMENU
1=PRINT
2=WYZIGEN KOP
3=PRINTER SETTING
4=HOE ITEMS PRINTEN
5=OMKEER RESET
6=OMKEREN BY PRINTEN
7=SELEKTEREN
8=GEEN SELEKTIE

UW KEUS?1
VANAF PAGNR?
```

4-8-83 PAG 1

CASSETTEARCHIEF
==============

GESORTEERD OP NAAM
SOORT VAN APP TOT APP

RNR	NAAM	BAND	TELLER	START	EIND	EXEC	SOORT	CAS	DATUM
1	DATPROC	202	2	2800	3C00	29	APP	V3	830610
2	INFOMASTER	202	60	8200	9FA5	84	APP	V3	830614

UW KEUS?8

UW KEUS?1
VANAF PAGNR?
4-8-83 PAG 1

CASSETTEARCHIEF
==============

GESORTEERD OP NAAM

RNR	NAAM	BAND	TELLER	START	EIND	EXEC	SOORT	CAS	DATUM
1	DATPROC	202	2	2800	3C00	29	APP	V3	830610
2	INFOMASTER	202	60	8200	9FA5	84	APP	V3	830614
3	SOURCEMAKER	201	2	4000	509B	42	SYSTEM	V3	830613

```
UW KEUS?4

   0        RNR
ITEM=J ?01

   1        NAAM
ITEM=J ?02

   2        BAND
ITEM=J ?03

   3        TELLER
ITEM=J ?N

   4        START
ITEM=J ?N

   5        EIND
ITEM=J ?N

   6        EXEC
ITEM=J ?N

   7        SOORT
ITEM=J ?N

   8        CAS
ITEM=J ?N

   9        DATUM
ITEM=J ?N

UW KEUS?1
VANAF PAGNR?
```

```
4-8-83                                              PAG    1
                       CASSETTEARCHIEF
                       ===============

GESORTEERD OP NAAM

 RNR NAAM               BAND

 1   DATPROC             202
 2   INFOMASTER          202
 3   SOURCEMAKER         201


 UW KEUS?1
 VANAF PAGNR?
```

```
UW KEUS?6

ITEM0=RNR              INT*2
ITEM1=NAAM             ASCII
ITEM2=BAND             INT*2
ITEM3=TELLER           INT*2
ITEM4=START            HEX*2
ITEM5=EIND             HEX*2
ITEM6=EXEC             HEX*2
ITEM7=SOORT            ASCII
ITEM8=CAS              ASCII
ITEM9=DATUM            INT*4
0E KOLOM = RNR
DIT WORDT NR?
1E KOLOM = NAAM
DIT WORDT NR?2
2E KOLOM = BAND
DIT WORDT NR?1
3E KOLOM = TELLER
DIT WORDT NR??
```

```
4-8-83                                              PAG    1
                       CASSETTEARCHIEF
                       ===============

GESORTEERD OP NAAM

 RNR BAND NAAM

 1    202 DATPROC
 2    202 INFOMASTER
 3    201 SOURCEMAKER

UW KEUS?
```

2.2 Forth Applicatie.
Voor de lezers onder u die Atom FORTH in hun bezit hebben, biedt deze applicatie enkele extra mogelijkheden t.o.v. de MC-routines uit het vorige hoofdstuk. Forth leent zich goed voor toepassingen als besturingen van bv. een PSG, het heeft de mogelijkheid met gedefinieerde woorden een complex van routines aan te roepen. Zo zal in deze applicatie uiteindelijk één woord gedefinieerd worden waarmee alle registerfuncties van de PSG aangesproken kunnen worden, nl. het woord "SOUND".
Ik ga er overigens van uit dat de FORTH bezitters ook op de hoogte zijn van de werking van FORTH, zodat hieronder slechts een opsomming volgt van de woord-definities.
Voordat de FORTH woorden gedefinieerd kunnen worden, moeten eerst weer een aantal basisroutines ge-"creeërd" worden (nl. om data in de registers te schrijven of uit de registers te lezen).

Deze routines zijn als volgt:

(write data to register):

```
LDA @ 112          initieer voor ontvangen van registernr.
STA # B800         zet 112 op poort B
LDA Ø,X            zet topwaarde van stack op poort A (reg.nr.)
STA # B801
LDA @ 96           zet 96 op poort B, initieer voor ontvangen van
STA # B800         data
LDA 2,X            zet voorlaatste waarde van stack op poort A
STA # B801         (registerinhoud)
LDA @ 64           zet 64 op poort B (Standby)
STA # B800
JMP NEXT           ga naar volgend woord
```

Deze routine verwacht op de FORTH stack een registernummer boven een registerinhoud.

(read data from register):

```
LDA @ 112          initieer voor ontvangen van registernr.
STA # B800         zet 112 op poort B
LDA Ø,X            zet topwaarde van stack op poort A (reg.nr.)
STA # B801
LDA @ 80           zet 80 op poort B, zet BC1,BC2 en BDIR voor
STA # B800         lezen van register
LDA @ Ø            maak poort A input
STA # B803
LDA # B801         lees de byte op poort A in accumulator
STA Ø,X            zet accumulator op stack (top)
LDA @ 64           zet 64 op poort B (standby)
STA # B800
LDA @ #FF          maak poort A weer output
STA # B803
JMP NEXT           ga naar volgend woord
```

Deze routine verwacht boven op de stack een registernummer en geeft de inhoud van dat register terug op de stack (top), op de plaats waar het registernummer stond.

(initialise ports)

```
LDA @ #FF          maak poort A output
STA # B803
LDA @ # F0         maak (1/2) poort B output
STA # B802
JMP NEXT           ga naar volgend woord
```

Deze routine initialiseert de poorten van de VIA.
De drie genoemde routines komen in de vorm van een CREATE in de FORTH applicatie.

Laad FORTH en typ de volgende definities in (wanneer men de applicatie op tape wil zetten laad dan eerst de editor):

```
HEX 8200 DP !
8200 FENCE !
CREATE SEP FFA9 , 8D C, B803 , F0A9 , 8D C, B802 , 4C C, 2842 ,
SMUDGE
CREATE RREG
70A9 , 8D C, B800 , 00B5 , 8D C, B801 ,
50A9 , 8D C, B800 , 00A9 , 8D C, B803 ,
AD C, B801 , 0095 , 40A9 , 8D C, B800 ,
FFA9 , 8D C, B803 , 4C C, 2842 ,
SMUDGE
CREATE WREG
70A9 , 8D C, B800 , 00B5 , 8D C, B801 ,
60A9 , 8D C, B800 , 02B5 , 8D C, B801 ,
40A9 , 8D C, B800 , 4C C, 2842 ,
SMUDGE
DECIMAL
: WRREG WREG 2DROP ;
: RESET SEP 14 0 DO 0 I WRREG LOOP 255 7 WRREG ;
: READ CR 16 0 DO ." REG: " I DUP . RREG . CR LOOP ;
: EV1 7 RREG 1 - 7 WRREG ;
: EV2 7 RREG 2 - 7 WRREG ;
: EV3 7 RREG 4 - 7 WRREG ;
: EV4 7 RREG 32 - 7 WRREG ;
: FRE1 256 /MOD 1 WRREG 0 WRREG ;
: FRE2 256 /MOD 3 WRREG 2 WRREG ;
: FRE3 256 /MOD 5 WRREG 4 WRREG ;
: FREN 6 WRREG ;
: AMP1 8 WRREG ;
: AMP2 9 WRREG ;
: AMP3 10 WRREG ;
: WAIT 8 * 0 DO LOOP ;
: ETIM 256 /MOD 12 WRREG 11 WRREG ;
: DV1 7 RREG 1 + 7 WRREG ;
: DV2 7 RREG 2 + 7 WRREG ;
: DV3 7 RREG 4 + 7 WRREG ;
: DV4 7 RREG 32 + 7 WRREG ;
: ATT DUP 4 * ETIM 13 13 WRREG WAIT ;
: REL DUP 4 * ETIM 1 13 WRREG WAIT ;
: ADR SWAP RCT ATT WAIT REL
: OF1 0 0 AMP1 FRE1 DV1 ;
: OF2 0 0 AMP2 FRE2 DV2 ;
: OF3 0 0 AMP3 FRE3 DV3 ;
```

```
: OF4 0 0 AMP3 FREN DV4 ;
: S1 EV1 FRE1 AMP1 ETIM DUP 16 = IF DROP ADR OF1 ELSE 13 WRREG
    DUP 0 > IF WAIT OF1 ELSE DROP THEN THEN ;
: S2 EV2 FRE2 AMP2 ETIM DUP 16 = IF DROP ADR OF2 ELSE 13 WRREG
    DUP 0 > IF WAIT OF2 ELSE DROP THEN THEN ;
: S3 EV3 FRE3 AMP3 ETIM DUP 16 = IF DROP ADR OF3 ELSE 13 WRREG
    DUP 0 > IF WAIT OF3 ELSE DROP THEN THEN ;
: S4 EV4 FREN AMP3 ETIM DUP 16 = IF DROP ADR OF4 ELSE 13 WRREG
    DUP 0 > IF WAIT OF4 ELSE DROP THEN THEN ;
: T3 DUP 3 = IF DROP S3 ELSE DROP S4 THEN ;
: T2 DUP 2 = IF DROP S2 ELSE T3 THEN ;
: SOUND DUP 1 = IF DROP T1 ELSE T2 THEN CR ." DONE " CR ;
```

Feitelijk is het FORTH woord SOUND het woord waar alle effecten mee bereikt kunnen worden. Echter is de manier van definiëren zo opgezet, dat ook alle andere woorden gebruikt kunnen worden om op andere wijze complexe geluidseffecten te creëeren.
De verklaring van de verschillende woorden is:

WRREG	write data in register
RESET	reset alle registers
READ	geef een dump van alle registerinhouden
EV1	enable "voice" op kanaal A
EV2	enable "voice" op kanaal B
EV3	enable "voice" op kanaal C
EV4	enable "noise" (ruis) op kanaal C
FRE1	verander de toonhoogte voice A
FRE2	verander de toonhoogte voice B
FRE3	verander de toonhoogt voice C
FREN	verander de ruis-grondfrequentie
AMP1	verander de amplitude kanaal A
AMP2	verander de amplitude kanaal B
AMP3	verander de amplitude kanaal C
WAIT	geeft een pauze van n maal 0,001 seconden (n op stack)
ETIM	verander de envelope timing waarde
DV1	disable voice op kanaal A
DV2	disable voice op kanaal B
DV3	disable voice op kanaal C
DV4	disable noise op kanaal C
ATT	geef een attack (gebruikt in ADR)
REL	geef een release (gebruikt in ADR)
ADR	ATTACK-DELAY-RELEASE, een(variable)ADR karakteristiek
OF1	voice A uit
OF2	voice B uit
OF3	voice C uit
OF4	noise C uit
S1	sound op kanaal A (voice)
S2	sound op kanaal B (voice)
S3	sound op kanaal C (voice)
S4	sound op kanaal C (noise)
T3	test of kanaal C (voice) of kanaal C (noise)
T2	test of kanaal B (voice) of T3
SOUND	forth woord voor klankopwekking (beschrijving zie onder)

Over het algemeen lijkt dit dus op hetgeen wat in de MC-routines wordt toegepast, de opbouw van een klank is onder te verdelen in een aantal basisfuncties.

Het woord "SOUND" kan op twee manieren gebruikt worden, in het eerste geval worden 6 parameters op de stack verwacht, in het tweede geval worden 8 parameters op de stack verwacht:

— top of stack

Mogelijkheid 1	Mogelijkheid 2
channel no	channel no
pitch	pitch
amplitude	amplitude
envelope tim.	envelope tim.
envelope shape	envelope shape
duration	release time
	delay time
	attack time

De keuze tussen mogelijkheid 1 of mogelijkheid 2 kan gemaakt worden door de envelope shape anders te kiezen (van 0-15 gaat FORTH uit van mogelijkheid 1 en bij het invullen van 16 voor de envelope shape gaat FORTH uit van de 2e mogelijkheid.).
Het gebruik is als volgt:

channel no.	:	1=voice A, 2=voice B, 3=voice C, 4=noise C (kleiner dan 0 of groter dan 4 is ook noise C)
pitch	:	een getal tussen 0 en 4095 voor voices een getal tussen 0 en 31 voor noise
amplitude	:	een getal tussen 0 en 15 is normaal volume-niveau (0= kanaal uit) 16 geeft envelope sturing van de amplitude van het kanaal.
Envelope timing	:	(d.i. de envelope timing waarde in R11 en R12) indien amplitude kleiner dan 16, geef dan 0.
Envelope shape	:	een getal tussen 0 en 15 (de normale envelope golfvormen volgens de tabel, d.w.z. de inhoud van R13) indien 16 als envelope shape, dan gaat FORTH uit van de parameters voor een ADR karakteristiek
Duration	:	alleen voor envelope-shapes kleiner dan 16, tijdsduur in eenheden van 0,001 s (1 ms) indien de waarde 0 wordt ingegeven blijft het kanaal continue open
Attack	:	alleen voor envelope shape =16, geeft een Attack fase (als onderdeel van een ADR karakteristiek) in eenheden van 0,001 s.
Delay	:	idem als attack, echter werkt als sustain (op maximaal amplitudeniveau), in eenheden van 0,001 s
Release	:	idem als attack, echter geeft een release fase in de ADR karakteristiek, eenheden in 0,001 s.

Enkele voorbeelden van het gebruik van SOUND:
(de parameters komen natuurlijk in omgekeerde volgorde te staan):

```
RESET
0 0 0 10 150 3 SOUND
0 0 0 10 151 2 SOUND
READ
(na enige tijd:)RESET
100 1000 800 16 0 16 200 1 SOUND
READ
```

Is het duidelijk hoe het werkt? probeer ook eens:

```
RESET
5000 8 300 16 150 1 SOUND (READ)
RESET
2500 10 500 16 15 4 SOUND (READ)
RESET
```

Er zijn wel enige restricties verbonden aan het gebruik van SOUND:
- gedurende de uitvoering van een SOUND is het niet mogelijk andere SOUND-s te programmeren, tenzij natuurlijk een continue klank is geprogrammeerd.
- men moet er rekening mee houden dat verschillende vormen van SOUND de registers met een andere inhoud achterlaten dan waarmee gestart werd (b.v. het enable register R7), wanneer dan een nieuwe SOUND volgt kunnen onverwachte effecten optreden. Overigens is dit in de meeste gevallen geen probleem.

Vanzelfsprekend kunnen naast SOUND andere woorden gedefinieerd worden om bijvoorbeeld meerdere kanalen met één SOUND aan te spreken.
Zoals gezegd kunnen alle andere woorden uit de applicatie ook gebruikt worden voor het opwekken van klanken, b.v:

```
RESET 100 101  FRE1 FRE2 EV1 EV2 16 16 AMP1 AMP2 500 ETIM 8 13 WRREG
(READ)
(RESET)
```

Maar ook nieuwe definities:

```
: NOISES EV4 10 AMP3 300 0 DO I FREN 20 WAIT LOOP OF4 ;
: KEYB EV1 16 AMP1 1000 ETIM BEGIN KEY ?ESC 2 * FRE1 1 13 WRREG UNTIL ;
```

(KEYB wacht op een toets en laat een toon horen met een frequentie 2 * de ascii waarde van de toets)

4. En nu?

In dit artikel zijn u een aantal mogelijkheden van de PSG (AY-3-8910) aangegeven, de software is slechts bedoeld om een indruk te geven van wat zoal mogelijk is. Ik kan mij echter voorstellen dat de kwaliteit van het geluidseffect of de muzikale kwaliteiten van de PSG volledig afhangen van de kwaliteit van de software. De PSG is in elk geval aardig te gebruiken in spelletjes (geluidseffecten) en daarvoor is een stukje software zoals beschreven onder 3.1 (MC routines) al redelijk geschikt .

Toch is de Atom SOUNDBOARD ook geschikt voor meer serieuze toepassingen. Een aantal software-ideeën zijn bijvoorbeeld:

- een basic muziek-editor, d.w.z. een programma waarmee stukken muziek gecomponeerd kunnen worden (dus invoeren, corrigeren en controleren). Ik zelf heb reeds een dergelijk programma geschreven (is wel ca. 14 Kbyte) waarbij de muziek in de vorm van normale notatie ingevoerd kan worden (toon en oktaaf), waarbij tonen tusseugevoegd en verwijderd kunnen worden. Zo'n programma vereist wel ingewikkelder software dan die welke in dit artikel is beschreven, zo zal je het programmagedeelte, waarbij de muziek gespeeld wordt, zgn. "realtime" moeten maken, omdat je anders alleen maar tonen krijgt, welke op hetzelfde moment inzetten of eindigen. (Met "realtime" bedoel ik dan dat je dus de mogelijkheid hebt de drie oscillators volledig onafhankelijk te gebruiken, dus ook op elk gewenst tijdstip een toon starten of beëindigen). In de praktijk zal daarom zo'n programma lang niet eenvoudig zijn.
- een "rhytme-box", m.b.v. de ruisgenerator en m.b.v. *één* of twee extra oscillators kan men een erg goede rhytmebox "maken". Hiertoe kan men een aantal subroutines maken, welke in een voorafbepaalde timing een ruiseffect laten horen (dus voor een cymbal een korte RELEASE (dus b.v. R13=0) en een ruis met veel hoge frequenties, voor een slag op de "snaredrum" ook een zeer korte RELEASE en ruis met hoge frequenties, maar dan gecombineerd met een oscillator welke op een lagere frequentie is ingesteld). Zo kan voor elk gewenst instrument, maar ook voor elk gewenst onderdeel van een rhytme-combinatie, een aparte subroutine geschreven worden welke het gewenste effect geeft. Door nu een centraal programma de verschillende subroutines aan te roepen (in een variabele volgorde en natuurlijk een variabele snelheid) kan men een goede imitatie van een rhytmebox verkrijgen. Aardiger wordt het, wanneer je een geprogrammeerde rhytme regelmatig afwisselt met bijvoorbeeld een "roffel" (een serie korte slagen op de snare-drum) of een slag op een cymbal (een langgerekte ruis), enfin de mogelijkheden zijn legio.

Een aantal hardware-mogelijkheden wil ik ook graag noemen:

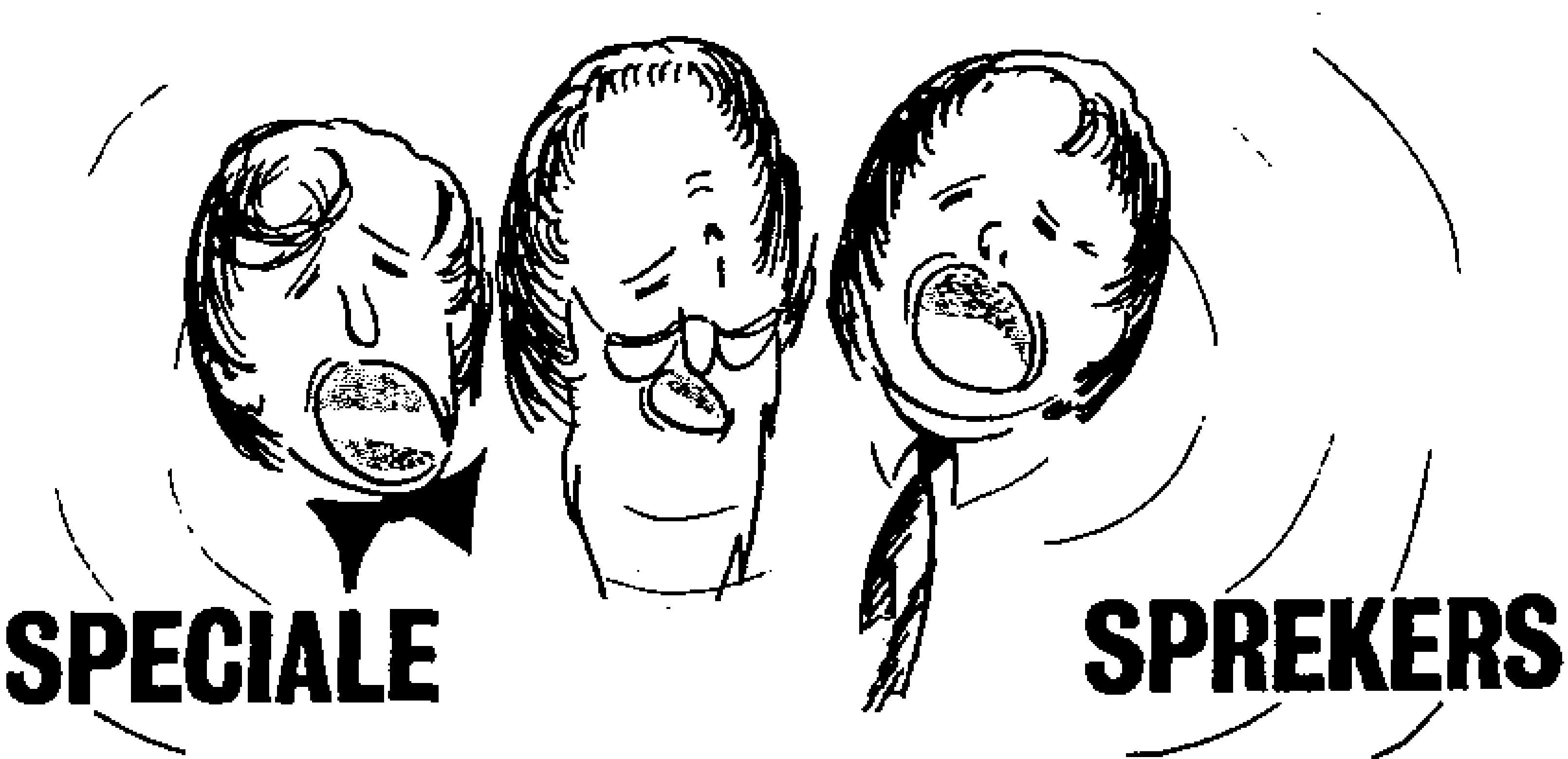
- het PSG uitgangssignaal is nog lang niet zo mooi als dat van de moderne synthesizers, echter kan men met extra electronica heel veel doen. Er zijn reeds schema's om een zogenaamde VCF achter de PSG te schakelen. Een VCF is een Voltage Controlled Filter, d.w.z. een spanningsgestuurd filter. Door zo'n filter met een afzonderlijk envelope generator te sturen, kan men muziekinstrumenten redelijk goed nabootsen. (Zoals wellicht bekend, kan men met een goed filter de golfvorm van de oscillator zeer sterk beïnvloeden). In de zeer moderne synthesizers werkt men overigens met digitale klankopwekking, d.w.z. dat men de uitgangs-golfvorm al volledig kan vaststellen (en naar wens nog kan beïnvloeden met een analoog filter). De digitale golfvormen kunnen dan in geheugens worden vastgelegd, op deze wijze ontstaan zogenaamde wave-tables, waaruit zeer snel een golfvorm gekozen kan worden. In de literatuuropgave neem ik een tweetal artikelen welke over zowel digitale klankopwekking gaan als over het koppelen van een filter aan de PSG('s). (D.z. de artikelen uit Computing Today).
- vanzelfsprekend kan men meerdere PSG's aan de Atom koppelen, tenslotte is de beschikking over twee nieuwe 8-bits poorten gekomen, waarmee betrekkelijk eenvoudig een tweede PSG bestuurd kan worden. Hierdoor krijgt men dus 6 oscillators, twee onafhankelijke envelope generators, etc.
- wanneer men echt serieus verder wil gaan met de PSG, dan is een combinatie

van meerdere PSG's, met extra randelectronica (bv. VCF's en "envelope" followers (dat zijn stukjes electronica welke een signaal afgeven afhankelijk van de envelope van het uitgangssignaal van de PSG, met zo'n envelope follower kun je dus de VCF weer sturen)) en een toetsenbord al een volledige computergestuurde synthesizer. Natuurlijk zal men dan het e.e.a. zelf moeten ontwikkelen (bijvoorbeeld de electronica om het toetsenbord aan de atom te koppelen).

Het vervolg op dit verhaal laat ik graag aan uw eigen fantasie over.

5. Literatuuropgave.

- Preliminary Information on the AY-3-891Ø, General Instruments (kan aangevraagd worden bij de chip).
- Sound out of the Atom, uit Acorn User, nummer onbekend.
- Uitbreiding van de speelcomputer, Elektuur september 1981, bldz. 38-44.
- Mozart Lives, Your Computer (Dice Waltz on the Spectrum), Vol.3 nr.3, bldz. 55-61.
- Micro Music, uit Computing Today, November 1982, bldz.45-48.
- Micro Music (p2), uit Computing Today,description of Micraulos on the Nascom, December 1982, bldz.37-4Ø.
- Forth Manual, Acorn Software.



Het raadsel van blz 85 in de vorige Acorn Nieuws is opgelost. De veranderingen op #C200,#C220....enz. (die overgens ook op #C080,#C0A0... enz kunnen voorkomen) worden veroorzaakt door een kleine onvolkomenheid in de JOS-box.
Ik zal bij het begin beginnen:
Het uitvoeradres van GRMOD is: #AE06. Als je dat met de STEP-commando ui de JOS-box volgt (kan alleen als GRMOD reeds is ingeschakeld) zul je zien dat hier alleen de schrijf- en leesvectoren worden veranderd in #ADE6 en #AD7E. Hierna wordt de accu geladen met waarde #C (=$12=scherm schoon) en wordt de processor via JSR #FFF4 naar de nieuwe schrijf routine gestuurd. Zie voor deze schrijf-routine STEP-lijst 1.
Eerst worden daar de normale zaken afgehandeld zoals: kijken of accu 2 of 3 is, (= kijken of printer aan of uit moet) opslag van alle registers in de stack, kijken of scherm aan moet -$6-, scherm uit moet =$#15- en kijken of scherm al uit staat.
Dan volgt JSR #ACE7. Deze routine "EXOR't" de cursor op huidige plaats met ?#E1. In deze routine wordt naar subroutine #AD5B gesprongen, welke het huidige adres van de cursor, of nog exacter, het bovenste streepje van de cursor uitrekent. En in deze routine komt de fout aan het licht.
Het adrs wordt berekend uit de inhoud van #E0 en #DF. Bij inschakelen van GRMOD staat echter in #DF nog het hoge byte van het oude adres van de cursor in TXMOD (normaal dus). Dit kan #80 of #81 zijn. Bij reeds ingeschakelde GRMOD staat in #DF het regelnummer van de cursor (0 t/m 15) en daar wordt keurig netjes het hoge byte, van het adres, van he bovenste streepje van de cursor uit berekend. (#80 t/m #96).
Zie voorbeeld STEP-lijst 2.
Bij eerste inschakeling van GRMOD wordt dus uitgegaan van de (foute) waarde #80 of #81. Na bewerking wordt dat #C0 of #C2. De vector op #5F en #60 wordt bij inschakeling van GRMOD in DIREKT-MODE #C080 of #C200, zie STEP-lijst 3.

```
Dan volgt: LDA(#5F),Y
           EOR #E1 (hierin staat nog #80.
           STA(#5F),Y
```

Dit is dus de verandering die je waarneemt in de interpreter. Register Y wordt nu steeds met #20 verhoogd en wanneer nodig wordt?#60 (hoge byte cursor adres) met 1 verhoogd. Alles bij elkaar worden er volgens mij 12 adresssen veranderd (en geen 20). Dit is niet met STEP te controleren omdat STEP zelf die geheugenplaaten ook gebruikt en veranderd.Tijdens het uitvoeren van scherm schoon, $12, wordt #E1 met waarde #FF gevuld.
Hiermede is het raadsel opgelost maar het probleem nog niet. Een vrij eenvoudige en afdoende oplossing is een schakelbare write-protect op het C000 blok (en eventueel op het #A000 en #D000 blok) aanbrengen en altijd opletten of deze ingeschakeld is bij overschakelen naar GRMOD. Je hebt dan tevens een goede beveiliging tegen allerlei ondeugdelijke software. Een lastige oplosstinf is: altijd het GRMOD commano vooraf te laten gaan door; ?#E1=00 (zie step-lijst 3), lastig want als je het een keer vergeet zit je weer met een verminkteinterpreter. Een definitieve oplossing is: zorgen dat de GRMOD routine (#AE0E) die 0 al in #E1 zet. Als je nu naar DISAS-lijst 1 kijkt zie je dat de laatste 4 bytes met 0 gevuld zijn. Deze worden volgens mij niet gebruikt en dus heb je hier mooi de ruimte om ?#E1 te veranderen. Hoe ik deze aanpassing gemaakt heb is te zien in DISAS-lijst 2. Eerst wordt het register X met 1 verhoogd en wordt dan 0. Dan wordt deze 0 in #e1 gezet, waarna de rest van de

normale routine volgt.
Je zou die 0 ook in #DF kunnen zetten maar dat geeft problemen bij het volgen van dat stukje routine met de step in TXMOD (omdat het cursor adres dan #0000 wordt)
Dus lijkt #E1 mij de beste oplossing. Na deze verandering heb ik alle JOS-box statements een voor een uitgeprobeerd om te kontroleren of die lege bytes echt niet gebruikt worden. Alles liep perfect en er zijn geen fouten of vastlopers voorgekomen. Die lege bytes werden dus echt nog niet gebruikt.

STEP-lijst 1 gemaakt in TXMOD.

```
AD6E:JSR #FEFB
AD71:PHP          0C 00 04 20 FE
AD72:PHA          0C 00 04 20 FD
AD73:CLD          0C 00 04 20 FD
AD74:STX #E4      0C 00 04 20 FD
AD76:STY #E5      0C 00 04 20 FD
AD78:JSR #ACBF    0C 00 04 20 FB
ACBF:CMP @#06     0C 00 04 21 FB
ACC1:BEQ #AC49    0C 00 04 21 FB
ACC3:CMP @#15     0C 00 04 A0 FB
ACC5:BEQ #AC4A    0C 00 04 A0 FB
ACC7:LDY #E0      0C 00 00 22 FB
ACC9:BMI #ACAA    0C 00 00 22 FB
ACCB:JSR #ACE7    0C 00 00 22 F9
ACE7:PHA          0C 00 00 22 F8
ACE8:JSR #AD5B    0C 00 00 22 F6
AD5B:LDA #E0      00 00 00 22 F6
AD5D:ASL          00 00 00 22 F6
AD5E:STA #5F      00 00 00 22 F6
AD60:LDA #DF      81 00 00 A0 F6
AD62:ASL          02 00 00 21 F6
AD63:ADC #DF      84 00 00 A0 F6
AD65:SEC          84 00 00 A1 F6
AD66:ROR          C2 00 00 A0 F6
AD67:STA #60      C2 00 00 A0 F6
AD69:ROR #5F      C2 00 00 22 F6
AD6B:LDY @#00     C2 00 00 22 F6
AD6D:RTS          C2 00 00 22 F8
ACEB:LDX @#02     C2 02 00 20 F8
ACED:STY #DE      C2 02 00 20 F8
ACEF:LDA (#5F),Y  B3 02 00 A0 F8
ACF1:EOR #E1      33 02 00 20 F8
ACF3:STA (#5F),Y  33 02 00 20 F8
ACF5:TYA          00 02 00 22 F8
ACF6:CLC          00 02 00 22 F8
ACF7:ADC @#20     20 02 00 20 F8
ACF9:TAY          20 02 20 20 F8
ACFA:CPY #DE      20 02 20 20 F8
ACFC:BNE #ACEF    20 02 20 20 F8
ACEF:LDA (#5F),Y  C2 02 20 A0 F8
```

STEPLIJST 2 in GRMOD.

```
ACE8:JSR #AD5B    0C 00 04 20 FD
AD5B:LDA #E0      00 00 04 22 FD
AD5D:ASL          00 00 04 22 FD
AD5E:STA #5F      00 00 04 22 FD
AD60:LDA #DF      06 00 04 20 FD
```

```
AD62:ASL           0C 00 04 20 FD
AD63:ADC #DF       14 00 04 20 FD
AD65:SEC           14 00 04 21 FD
AD66:ROR           8A 00 04 A0 FD
AD67:STA #60       8A 00 04 A0 FD
AD69:ROR #5F       8A 00 04 22 FD
AD6B:LDY @#00      8A 00 00 22 FD
AD6D:RTS           8A 00 00 22 FF
```

STEPLIJST 3 in TXMOD.

```
ACE8:JSR #AD5B     0C 00 04 20 FD
AD5B:LDA #E0       00 00 04 22 FD
AD5D:ASL           00 00 04 22 FD
AD5E:STA #5F       00 00 04 22 FD
AD60:LDA #DF       80 00 04 A0 FD
AD62:ASL           00 00 04 23 FD
AD63:ADC #DF       82 00 04 A0 FD
AD65:SEC           82 00 04 A1 FD
AD66:ROR           C1 00 04 A0 FD
AD67:STA #60       C1 00 04 A0 FD
AD69:ROR #5F       C1 00 04 22 FD
AD6B:LDY @#00      C1 00 00 22 FD
AD6D:RTS           C1 00 00 22 FF
ACEB:LDX @#02      C1 02 00 20 FF
ACED:STY #DE       C1 02 00 20 FF
ACEF:LDA (#5F),Y   65 02 00 20 FF
ACF1:EOR #E1       E5 02 00 A0 FF
ACF3:STA (#5F),Y   E5 02 00 A0 FF
```

DISAS-LIJST 1 orginele JOS-BOX.

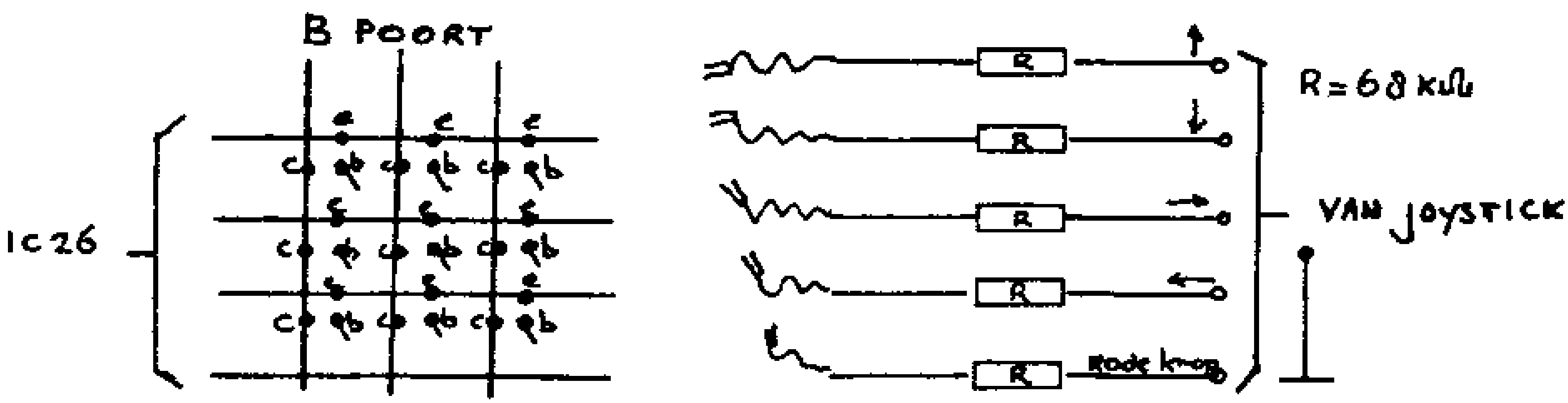
```
AE06: A0 08        LDY @#08
AE08: A2 03        LDX @#03
AE0A: B9 13 AF     LDA #AF13,Y
AE0D: 9D 08 02     STA #0208,X
AE10: 88           DEY
AE11: CA           DEX
AE12: 10 F6        BPL #AE0A
AE14: A9 0C        LDA @#0C
AE16: 20 F4 FF     JSR #FFF4
AE19: 4C AD F6     JMP #F6AD
AE1C: 00           BRK
AE1D: 00           BRK
AE1E: 00           BRK
AE1F: 00           BRK
```

DISAS-LIJST 2 met de verandering.

```
AE06: A0 08        LDY @#08
AE08: A2 03        LDX @#03
AE0A: B9 13 AF     LDA #AF13,Y
AE0D: 9D 08 02     STA #0208,X
AE10: 88           DEY
AE11: CA           DEX
AE12: 10 F6        BPL #AE0A
AE14: E8           INX
AE15: 86 E1        STX #E1
AE17: A9 0C        LDA @#0C
AE19: 20 F4 FF     JSR #FFF4
AE1C: 4C AD F6     JMP #F6AD
```

Geachte hobbygenoten.

Naar aanleiding v/e artikel in AN6 "83 blz.60, kan ik het volgende toevoegen.
De matrix naar buiten brengen, heb ik al lang geleden gedaan en tot volle tevredenheid. Alleen het aansluiten van een Joystick stuit op een practisch Probleem, aangezien deze een gemeenschappelyke retourleiding (GND) heeft.
Oplossing zou kunnen zyn om alle schakelkontakten apart naar buiten te voeren. We hebben dan een tienaderige kabel nodig en dat is lastig en geeft snel verwarring met wat aan wat moet.
Beter is het om de matix op een printje te maken en met transistoren te schakelen. Gebruik hiervoor PNP's silicium met de E. aan de B-Poort en de C. aan de uitgangen van IC26. De basis komt aan een printpennetje waarop U straks het stekertje komende v/d Joystick prikt. De Joystick kan zo ongewyzigd blyven en er zyn maar vyf stekertjes nodig om op de desgewenste pennen te prikken. In serie met de vyf lynen v/d Joystick moeten wel weerstanden van ca. 68Kohm geschakeld worden. (Niet in de retour!)
U hoeft natuurlyk niet de gehele matrix vol met transistoren te zetten, doch alleen die plaatsen welke U nodig denkt te hebben.
Een klein schematje zal eea verder verduidelyken. Het is maar een klein deel v/d matrix, maar geldt dus voor elke plaats in de matrix.



Gaan we even verder naar blz.88 van AN6 "83.
Myn schakelkaart werkt al heel lang met 1 defecte DEAC cel waarvan de spanning ongeveer 1,2 Volt bedraagt!!!
Volgens het betreffende artikel zou dit niet kunnen. Ik kan U echter verzekeren dat de KAART perfect werkt, zelfs met nog lagere spanning. Mogelyk dat de stoorgevoeligheid toeneemt, alhoewel myn KAART ook wat dat betreft al heel wat op z'n kop heeft gehad. Veel te hoge spanning (10V), Plus en min omdraaien enz. Alles bleef keurig en foutloos in het geheugen.
Het is maar dat U zich niet onnodig zorgen maakt.

P-CHARME UITBREIDEN

Zoals beschreven in de handleiding van P-Charme is het nu zeer gemakkelijk geworden om extra statements te programmeren.
Voor het aanmaken van de tabel moet echter nog het een en ander worden ge-poked. Om het inladen van de in Acorn Nieuws afgedrukte statements te vergemakkelijken is een standaard manier ontworpen die uitblinkt in eenvoud. In eerste instantie had ik hiervoor een lang programma geschreven, maar omdat dit een 'intelligent' (?) stukje software was, bleek al snel dat m.b.v. een beetje eigen intelligentie het allemaal veel korter kon. Aan de hand van een voorbeeld zal een en ander vlug duidelijk worden:

Stel we willen een nieuwe tabel met statements maken op #6800. Een tabel mag op z'n hoogst 256 bytes lang zijn, dus we kunnen veilig beginnen met assembleren op #6900 (als we zeker weten dat onze tabel ook in de toekomst nooit zo lang wordt, dan mogen we natuurlijk ook eerder beginnen met assembleren).

We beginnen met het aanmaken van een lege tabel (een tabel die dus wel door P-Charme herkend zou kunnen worden, maar die nog geen statements bevat). Zo'n tabel ziet er alsvolgt uit:

```
6800  FF
6801  B3
6802  C6
6803  80      (tabel eindigt altijd met #80)
```

We kunnen die tabel dus gemakkelijk maken met:

```
!#6800=#80C6B3FF
```

Om nu de gepubliceerde statements te kunnen assembleren moeten we nog twee dingen doen:

a) de variabele T wordt gelijk gemaakt aan het eind-adres van de tabel. In het begin is dat dus #6803.

```
T=#6803
```

b) de variabele A wordt gebruikt om de assembler te vertellen waar de statements moeten komen. Voor het eerste statement kiezen we nu:

```
A=#6900
```

Vervolgens runnen we achter elkaar de programma's van de statements die we in onze tabel willen plaatsen. Het begin van al die programma's is steeds ongeveer gelijk en daar wordt onder andere de tabel gevuld

en worden de waarden van de variabelen T en A met de juiste waarde opgehoogd zodat T weer wijst naar het einde van de tabel en A wijst naar het eerste vrije byte achter het geassembleerde statement. Bovendien wordt het adres van dat vrije byte opgeslagen achter het einde van de tabel. Op deze manier kunnen we dat later weer gemakkelijk terugvinden.

Met een HEX-dump kunnen gemakkelijk het byte #80 opzoeken dat het einde van een tabel aangeeft. Maken we dan T gelijk aan het adres van dat byte dan kunnen we het vrije assembler adres gemakkelijk toekennen aan de variable A met:

```
A=T!1
```

Hierdoor kunnen we ook de statements uit de volgende Acorn Nieuws gemakkelijk in de tabel plaatsen.

Om de P-Charme interpretor wijs te maken dat er een tabel met statements op 6800 staat moeten we nog op #3FC het hoge byte van de tabel poken:

```
?#3FC=#68
```

Voor diegenen die de statements op de 16K-kaart zetten is dit nog het enige dat gepoked moet worden na het aanzetten van de ATOM. De machines met een ingebouwde bootstrap kunnen natuurlijk zo worden gewijzigd dat dit automautisch gebeurt. Voor de mensen zonder bootstarp, maar met 16K-kaart is het misschien handig om ergens op slot een klein programmaatje te plaatsen met de naam INIT (m.b.v. het PROGRAM statement) dat er ongeveer zo uit zou kunnen zien:

```
10 PROGRAM INIT
20
30 ?#3FC=#68
40 ?18=#29;LINK #C2B2
50 END
```

Het 'statement' (?) INIT zorgt er dan voor dat je over de extra statements kunt beschikken. Uiteraard kunnen in het INIT programma ook andere zaken worden geinitialiseerd (de DOS kan bijvoorbeeld worden ingeschakeld).

Bovenstaande is voor twee groepen van gebruikers interessant; ten eerste voor die mensen die zelf geen statements schrijven, maar wel de statements uit Acorn Nieuws willen gebruiken en ten tweede voor diegenene die wel statements schrijven. Voor hen is het handig om te weten dat een zeer grote groep leden de beschikking heeft over P-Charme en dat de hiervoor genoemde eerste groep gebruikers er bij gebaat is dat gepubliceerde statements makkelijk te gebruiken zijn. Het is dus aan te bevelen (maar geheel niet verplicht of zo) om statements zo te schrijven dat ze geassembleerd kunnen worden met behulp van de variabelen A en T.

Veel succes,

FRANS

MORE-DIRECTORY

Hoewel in het gewone gebruik van de disk-drive het maximale aantal files van 31 per schijf, voldoende mag heten, werkt het bijzonder storend met bijvoorbeeld FORTH. Ook heb ik van enkele spelletjes fanaten begrepen dat het bijzonder handig zou zijn om een groter aantal te hebben, bijv. voor het opslaan van diverse Data-files.

Wel met het volgende programma kan dat. In totaal kunnen hiermee 174 files (verdeeld over 6 pagina's met ieder 16 K schijfruimte ter beschikking) per schijf bewaard worden. In principe werkt een en ander als volgt:

Middels de instructie *P N wordt allereerst de complete routine (onder file naam P) geladen op #2200-#2400 en gestart op #2200. Hoewel dit gebied in principe gereserveerd is voor de verwerking van randomfiles, is dit geen probleem, omdat bij wisseling van directory alle random files gesloten moeten zijn. Bovendien laat de beschikbare 16K per pagina slechts 1 randomfile toe. Na controle en interpretatie van de parameter N wordt de oude directory opgeslagen op de schijf. Vervolgens wordt de gewenste directory-page N van z'n opslagplaats gehaald en als nieuwe directory geinstaleerd. De parameter N mag zowel een getal als een variabele zijn, mits 0<N<7. Bovendien werkt een en ander zowel in de directe mode, vanaf het toetsenbord, als vanuit een lopend Basic-programma.

Dus 6 directory's met ieder 29 files. De resterende 2 files per directory (in de routine omschreven als "page") zijn nodig a) voor de sturende routine, en b) om de op dat moment niet beschikbare andere files te beschermen. Een en ander wordt uitgevoerdt door de routines beschreven in regels 60-250.

De rest van het programma (dwz. nadat het geassembleerd is) zorgt voor het initialiseren van de totaal routine (regels 290-590). Dwz. na het intikken van het programma kan men een reeds geformatteerde (lege) schijf voorzien van 174 files verdeeld over 6 pages, door te linken naar LL0. Het zorgt dan voor de opmaak van de 6 pages plus de opslag van de gehele routine op de schijf. Is de routine eenmaal aanwezig, dan kunnen andere schijven ook voorzien worden door de instructie *P 0. Het gebruik van andere parameter waarden dan 0-6 zal resulteren in Error 189 : Illegal Quantity.

Voor de rest van de werking van het programma verwijs ik naar het commentaar in het programma.

NB. Bij het interpreteren van de parameter na invoer van de instructie *P N wordt als parameter alleen de laatste letter/cijfer voor de <CR> geevalueerd. Dus na invoer van *P KOLDER3, zal page 3 ingeladen worden.

Veel succes ermee,

Ruurd van der Zee.

```
UP-TO-YOU     DRIVE 0 QUAL

PAGE 1
 1      MOREDIR        #P
        TEST           moredi3
```

```
   10 PROGRAM MORE DIR
   20 DIM XX1,EE2,LL6;FOR I=0TO12;XXI=-1;NEXT
   30 PRINT $21
   40 FOR N=1 TO 2;P=#2200
   50 [
   60 \ INTERPRETE PARAMETER
   70    JSR #E226;DEC #03;DEC #03;JSR #CF3E;CPY @#00;BCC XX0
   80    BEQ LL0;CPY @#07;BCC XX1
   90:XX0;JMP #C46B \ ERROR 109
  100:XX1;STY #23C3
  110
  120 \ SAVE OLD DIRECTORY
  130    LDA #2107;JSR #E0FA;ADC @#01;BNE EE1;LDA @#08
  140:EE1;STA #ED;LDA #2106;BEQ EE2;DEC #ED;DEC #ED
  150:EE2;STA #EC;JSR #E705;JSR #E74D;JSR #E226
  160
  170 \ LOAD NEW DIRECTORY
  180    JSR #E75B;LDY #23C3;LDA #23C3,Y;STA #EC;LDA #23C9,Y
  190    JSR #E703;JSR #E73C;JSR #E226
  200
  210 \ REPLACE DISK DIRECTORY WITH NEW PAGE
  220    JSR #E74A
  230    RTS
  240]
  250
  260 REM FORMAT
  270[:LL0;JSR #E016;]
  280 $P="DO YOU REALLY WANT TO MULTIPLY  DIRECTORIES ON DRI
VE "
  290 P=P+LEN P
  300[
  310    LDA #EE;JSR #F80B;JSR #E016;]
  320 $P=" ?";P=P+2
  330[
  340    NOP;JSR #FFE6;CMP @#59;BNE LL6
  350    JSR #FFE6;CMP @#45;BNE LL6;JSR #FFE6;CMP @#53;BNE LL
6
  360
  370 \ PREPARE 6 DIR-PAGES ON DISK
  380    JSR #E75B;LDY @#17
  390:LL3
  400    LDA #23D0,Y;STA #2000,Y;LDA #23E8,Y;STA #2100,Y
  410    DEY;BPL LL3;LDY @#06
  420
  430:LL4
  440    STY #23C3;LDA #23C3,Y;STA #EC;LDA #23C9,Y
  450    JSR #E703;JSR #E74D;JSR #E226;LDY #23C3;DEY;BEQ LL5
  460    LDA #23AE,Y;STA #2106;LDA #23B3,Y;STA #2107;LDA #23B
8,Y
  470    STA #210D;LDA #23BD,Y;STA #210E;DEC #200E;JMP LL4
  480
```

```
 490:LL5
 500    JSR #E6A7  \ FILL DISK-DIR WITH PAGE 1
 510
 520 \ SAVE THIS ROUTINE
 530    LDA @#22;STA #9D;LDA @#00
 540    STA #9C;STA #A0;STA #A2;LDA @#02;STA #A1;STA #A3
 550    JSR #E6B0
 560:LL6
 570    JMP #FFED
 580]
 590
 600 PRINT $6
 610 NEXT;PRINT $3
 620 FOR I=P TO #23FF STEP 4;!I=#EAEAEAEA;NEXT
 630
 640 REM TABEL DATA TRACK/SECTOR FOR EACH DIR-PAGE
 650 P=#23C3;RESTORE 700
 660 FOR I=1 TO 6
 670 READ T,S;P?I=T;P?(I+6)=S
 680 NEXT
 690
 700 DATA 0,4;REM TRACK,SECTOR
 710 DATA 0,6
 720 DATA 0,8
 730 DATA 1,0
 740 DATA 1,2
 750 DATA 1,4
 760
 770 REM TABEL DATA FIRST DIRECTORY (PAGE 1)
 780 P=#23D0
 790 RESTORE 840
 800 FOR I=0 TO #2F
 810 READ X;P?I=X
 820 NEXT
 830
 840 DATA #55,#50,#2D,#54,#4F,#2D,#59,#4F
 850 DATA #0D,#50,#41,#47,#45,#20,#36,#00
 860 DATA #50,#20,#20,#20,#20,#20,#20,#A0
 870 DATA #55,#20,#20,#20,#20,#10,#01,#90
 880 DATA #00,#00,#00,#00,#00,#0C,#10,#04
 890 DATA #00,#22,#00,#22,#00,#02,#00,#02
 900
 910 REM TABEL DATA DIFFERENT FOR EACH PAGE
 920 RESTORE 1010
 930 P=#23AE
 940 FOR I=0 TO 19 STEP 5
 950    FOR J=1 TO 5
 960    READ X;P?(I+J)=X
 970    NEXT J
 980 NEXT I
 990 END
1000
1010 DATA #00,#00,#00,#01,#01
1020 DATA #50,#90,#D0,#10,#50
1030 DATA #0C,#4C,#8C,#CC,#0C
1040 DATA #00,#00,#00,#00,#10
```

SUPER DRAW

Gebruiksaanwijzing SUPERDRAW

disk/cassette versie april 1984
* beschrijving :

"SUPERDRAW" is een programma voor de Acorn Atom dat het mogelijk maakt, mits voldoende
geheugen voorhanden is (zie onder het kopje "geheugengebruik"), in dit geheugen een grafisch beeld op te bouwen met een oplossend vermogen van 512 (hor.) * 384 (vert.) punten.
In principe is hiervan niets zichtbaar op uw scherm.
Het programma echter biedt de faciliteit om een gedeelte van de afbeelding naar het schermgeheugen met een capaciteit van 256 * 192 punten te kopieren zodat precies een kwart van de afbeelding te zien is.

Dit scherm of venster is in stappen van 8 punten in de vier richtingen over de afbeelding te schuiven zodat elk deel van de afbeelding bekeken kan worden. Dit principe noemt men 'windowing' (zie fig.1).
Dit extra scherm nu biedt de mogelijkheid om een aantal bewerkingen uit te voeren tussen dit scherm en het onderliggende deel van de afbeelding.
Zo vindt het inkleuren van beeldgedeeltes plaats op het scherm zonder de afbeelding te veranderen: mocht het resultaat tegenvallen dan is de oorspronkelijke situatie eenvoudig te herstellen.
Verder kan men een overzicht van de gehele afbeelding krijgen door deze te verkleinen naar het scherm.
Overige mogelijkheden zijn het punt voor punt tekenen van lijnen in 8 richtingen (met REPEAT 60 punten per seconde), het trekken van lijnen, het spiegelen van scherm of afbeelding, het vergroten van de scherminhoud naar de afbeelding, het inkleuren van contouren met 1 van de 24 grijstinten (zie fig.2), het aanbrengen van een raster, een kader, het inverteren van de afbeelding, invoeren van tekst, laden/saven van de afbeelding of een enkel scherm.
Vaardigheid dient ook hier verkregen te worden door oefening: het voordeel is dat u het resultaat van vrijwel iedere bewerking direkt op uw scherm ziet verschijnen.
Voor op-, aanmerkingen en suggesties houd ik mij van harte aanbevolen.

Rob van Dort
M53
7273 TA Haarlo

* laden : DATA met *LOAD "DATA"; *RUN "SPRDRAW"

* geheugengebruik :

programma (machinetaal)	9C00-AFB0
data characters	1C00-2000
stackruimte inkleurroutine	021C-03FC
opslag afbeelding	2000-8000
scherm of venster	8000-9800
tijdelijke opslagruimte	9800-9C00
opstart-adres (SSSS)	9C00

* opstarten : (na onderbreking) LINK #SSSS ; antwoord op eerste vraag JA of anders

* instructies :

DEELl 1: CONTROLFUNCTIES

- ctrl-@ : naar menu voor bediening diskdrive of cassetterecorder
- ctrl-L ** : wis afbeelding, cursor + scherm linksonder
- ctrl-A : afbeelding 8 posities naar links t.o.v. scherm
- ctrl-S : idem , naar rechts
- ctrl-W : idem , naar boven
- ctrl-Z : idem , naar beneden
- ctrl-H : cursor en scherm naar HOME-positie linksonder
- ctrl-I : inverteer afbeelding
- ctrl-G ** : vergroot het scherm 2* lineair naar de afbeelding
- ctrl-M ** : kopieer het scherm naar het onderliggend deel van de afbeelding
- ctrl-T : maak invoeren van tekst mogelijk subkommando's:
  ctrl-J : cursor 10 omlaag
  ctrl-K : cursor 10 omhoog
  RETURN : nieuwe regel
  DELETE : verwijder laatst ingevoerde teken
  ctrl-T : verlaat tekstinvoer mode
- ctrl-E : verwissel scherm / onderliggend gedeelte van de afbeelding (exchange)
- ctrl-D ** : dump afbeelding naar printer (zie NOOT 2)
- ctrl-F : scherm aan: het scherm toont precies wat 'eronder' ligt
- ctrl-U : scherm uit: het scherm wordt ontkoppeld van de afbeelding
- ctrl-K : teken een kader om de afbeelding
- ctrl-N ** : AND scherm / afbeelding
- ctrl-O ** : OR scherm / afbeelding
- ctrl-X ** : EXOR scherm / afbeelding
- ctrl-P : positioneer cursor in midden van scherm
- ctrl-Q ** : quit 'SUPERDRAW' naar lege Basic tekstruimte #8200
- ctrl-R ** : EXOR de afbeelding met een raster met maas 8
- ctrl-V : verklein de afbeelding naar het scherm
- ctrl-Y gevolgd door een cijfer 0 t/m 9: 'verdun' het scherm
- ctrl-C gevolgd door een letter A t/m W of a t/m w : vul de contour op het scherm waarbinnen de cursor zich bevindt met het gevraagde grijsraster

DEEL 2 : LETTERFUNCTIES

- A : cursor 1 positie naar links
- S : idem naar rechts
- W : idem naar boven
- Z : idem naar beneden
- a : cursor 10 posities naar links
- s : idem naar rechts
- w : idem naar boven
- z : idem naar beneden
- [: cursor links op regel
-] : cursor rechts op regel
- @/SHIFT-@ : inverteer punt onder cursor
- G : cursor naar links en reset punt onder cursor
- T : idem naar linksboven
- Y : idem naar boven
- U : idem naar rechtsboven
- H : idem naar rechts
- N : idem naar rechtsonder
- B : idem naar beneden
- V : idem naar linksonder
- g : cursor naar links en set punt onder cursor
- t : idem naar linksboven
- y : idem naar boven
- u : idem naar rechtsboven
- h : idem naar rechts
- n : idem naar rechtsonder
- b : idem naar beneden
- v : idem naar linksonder
- / : definieer startpositie van te trekken lijn
- \5 : trek witte lijn naar cursor (PLOT 5)
- \6 : inverteer lijn naar cursor (PLOT 6)
- \7 : trek zwarte lijn naar cursor (PLOT 7)
- > : roteer regel waarop cursor staat 1 positie naar rechts
- < : roteer regel waarop cursor staat 1 positie naar links
- ↑ : spiegel scherm (alleen versie A)
- SHIFT-↑ : spiegel afbeelding (alleen versie A)
- 0 : knipper een blok rond de cursor
- 1 : scherm linksonder
- 2 : scherm rechtsonder
- 3 : scherm linksboven
- 4 : scherm rechtsboven

** betekent : de betreffende toetsaanslag 2 keer uitvoeren en afsluiten met RETURN (= ctrl-M)

* beschrijving menu :
- aanroep met ctrl-@;
- een commando moet worden ingevoerd zonder de * (bijv. CAT ipv *CAT);
- indien een fout opgetreden is (disk vol, file niet gevonden of beschermd, etc.), wordt het menu opnieuw gegeven;
- het systeem wacht ook bij gebruik van diskdrive tot de uitvoering van een opdracht voltooid is zodat beschadiging van file en/of afbeelding voorkomen wordt.

NOOT 1 : met ESCAPE is iedere foutieve invoer te onderbreken

NOOT 2 : op ctrl-D ** wordt een JSR #SSSS+#0017 uitgevoerd alwaar een JMP behoort te staan naar de start van een door uzelf gemaakte printerdumproutine. Verander daartoe de bytes #SSSS+#0018 en #SSSS+#0019 in het gewenste adres.

NOOT 3 : de cursor geeft ook aan waar in de tekstinvoer mode het eerstvolgende teken verschijnt.

NOOT 4 : is het resultaat na inkleuren niet het bedoelde, dan is de oorspronkelijke situatie eenvoudig te herstellen met ctrl-F; het resultaat is pas definitief na ctrl-M **; het inkleuren is te onderbreken met de SHIFT-toets.

NOOT 5 : tijdens het inkleuren van zeer complexe figuren kan het voorkomen dat het proces voortijdig stopt en 3 piepjes geeft: dit betekent dat de stackruimte voor de inkleurroutine vol is. Dikwijls is door een andere keuze van de startpositie (de cursor)
dit probleem op te lossen. De beste startpositie bij zulke figuren is zodanig dat links en rechts van de cursor ongeveer evenveel aftakkingen zitten. Bij 'normale' figuren hoeft u zich hier geen zorgen om te maken.

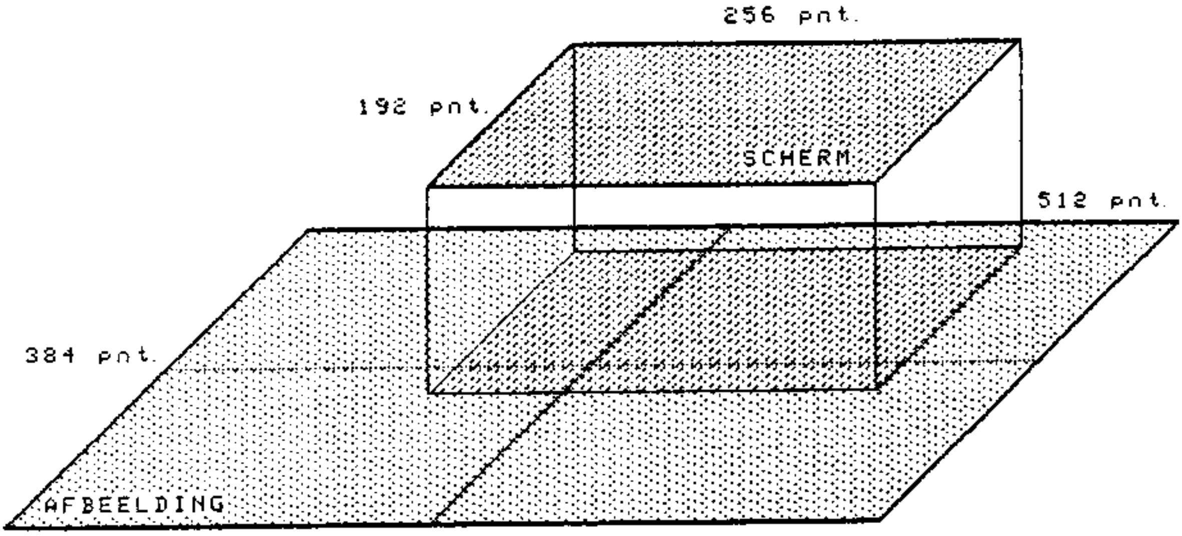
NOOT 6 : de gebruiker kan beschikken over de PLOT-routine van SUPERDRAW. LINK #SSSS+#000A is een 'CLEAR 5' : wist #2000 - #7FFF
en vult het adres van de routine in op #3FE,#3FF. Alle volgende BASIC MOVE-, DRAW- en PLOT opdrachten vinden nu plaats op de grote afbeelding. Zorg wel dat het programma dat het tekenwerk verricht op een veilige plaats staat !!! (bijvoorbeeld vanaf #8200)
ZET VOOR ALLE ZEKERHEID UW TEKENPROGRAMMA OP TAPE / SCHIJF VOORDAT U 'RUNT'

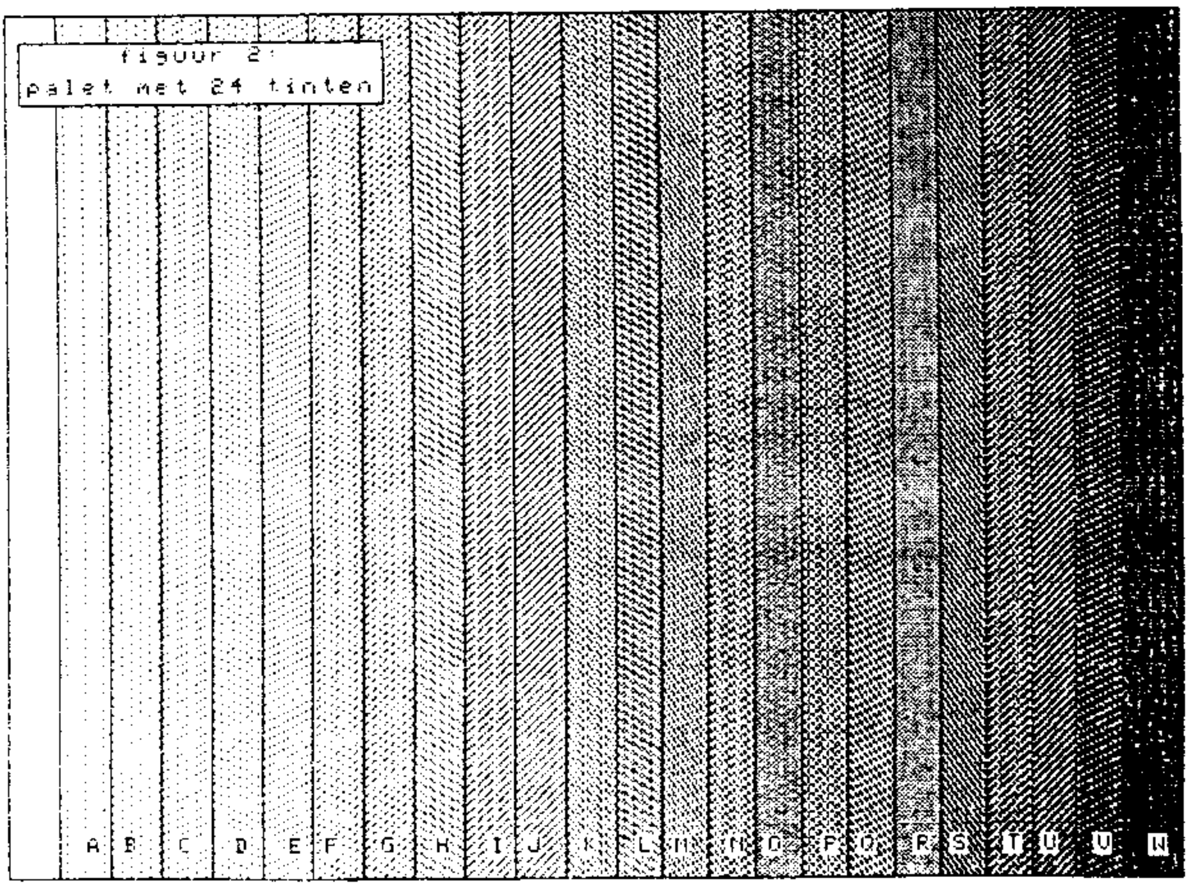
NOOT 7 : wees er op bedacht dat bij 'normaal' gebruik van de disk-drive het gebied #2000-#2200 beschadigd wordt. Onder 'SUPERDRAW' is de afbeelding hiertegen beschermd.

NOOT 8 : bij verlaten van 'SUPERDRAW' wordt u neergezet op BASIC tekstruimte #82. Dit is te veranderen door op adres #SSSS+#001B de gewenste tekstruimte in te vullen.

NOOT 9 : na een inkleur-operatie is de cursorpositie voor het trekken van lijnen gewist ! opnieuw zetten met ...

figuur 1
principe van 'windowing'





figuur 2:
palet met 24 tinten

```
  0 REM 3-DIM. PLOT 23-3-'84
 10 REM DEMONSTRATIE 'SUPERDRAW'
 20 REM DRIE-DIMENSIONALE PLOT
 30 REM 'SUPERDRAW' MOET GELADEN ZIJN !
 40 REM LAAD DIT PROGRAMMA VANAF #9800 (>#8200) !!!
 50 REM ZIE OOK ATOMIC THEORY & PRACTICE P.83
 60 REM ROB VAN DORT
 70 PRINT $12$10$10"     1.5 MINUUT GEDULD S.V.P.""
 80 L=50;M=80;N=16;REM COORDINATEN TOESCHOUWER
 90 LINK #9C0A ;REM 'CLEAR 5'
100 Z=4
110
120 S=L*L+M*M
130 R=SQR(S)
140 T=S+N*N
150 Q=SQR(T)
160
170 C=256;D=192;REM COORDINATEN MIDDELPUNT SCHERM
180
190 FOR U=-40 TO 40 STEP 4
200    V=-40;GOSUB c;GOSUB b
210    FOR V=-38 TO 40 STEP 2
220       GOSUB c;GOSUB a
230    NEXT V
240 NEXT U
250
260 FOR V=-40 TO 40 STEP 4
270    U=-40;GOSUB c;GOSUB b
280    FOR U =-38 TO 40 STEP 2
290       GOSUB c;GOSUB a
300    NEXT U
310 NEXT V
320 LINK #9C00
330 END
340
350aZ=5
360bO=R*(T-U*L-V*M-W*N)
370 X=C+T*(V*L-U*M)*4/O
380 Y=D+3*Q*(W*S-N*(U*L+V*M))/O
390 PLOT Z,X,Y;Z=4
400 RETURN
410
420cW=600/(10+U*U+V*V)-10;R.
```

Een inktjet-printer heeft als grote verschil met de normaal door ons gebruikte printer, dat hij geen contact maakt met het papier. Er wordt dus niet met matrix naalden of met lettertekens via een lint op papiergedrukt.
Daarentegen spuit de inkt-jet printer de inkt via een glazen staaf op het papier. Het grote voordeel hiervan is de geruisloosheid. Ook is men niet meer afhankelijk van de vaste beeld-dragers of van het voorgeschreven matrix patroon. Hierdoor heeft men de beschikking over de maksimale mogelijkheden in het grafisch gebeuren.

Een inktjet-printer wordt op de markt gebracht door Olivetti. De technische gegevens van deze printer luiden als volgt:

TYPE: PR 2300
DRY INKT YET PRINTER
SCHRIJFT BIDIRECTIONEEL
SNELHEID: 50 TOT 130 REGELS PER MINUUT.
SCHRIJFT MET ISØ 96 KARAKTERS (7 * 7 MATRIC)
AFHANKELIJK VAN KEUZE: 80, 97 OF 147 KARAKTERS OP EEN REGEL.
WERKT MET LOSSE VELLEN PAPIER OF ROL VAN 216 mm OF
CONTINU DOORVOER VAN PAPIER (PIN FEED).
GEWICHT: 5,2 kg
AFMETINGEN: 384mm * 114mm * 264mm.
GEEN AIR CONDITIONING VEREIST.
ADVIESVERKOOPPRIJS F. 1.695,00 excl. B.T.W.

Leden van de ACORN COMPUTER CLUB kunnen deze printer kopen voor de ledenprijs van F. 900,00 exlc. B.T.W.
Bestellingen hiertoe dienen UITSLUITEND SCHRIFTELIJK te worden gedaan bij:
Dhr. W. Bonewald.
Goudrenetstraat 6,
5632 NV Eindhoven.

Door Olivetti wordt daarna een factuur naar de aanvrager gestuurd. De levering zal geschieden na betaling franko.

```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk.
qrstuvwxyz{|}~!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]
cdefghijklmnopqrstuvwxyz{|}~!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNO
UVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~!"#$%&'()*+,-./0123456789:;<=>?@AB
GHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~!"#$%&'()*+,-./01234
9:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~!"#$%&
+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstu
{|}~!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefgh
mnopqrstuvwxyz{|}~!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
_`abcdefghijklmnopqrstuvwxyz{|}~!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKL
QRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~!"#$%&'()*+,-./0123456789:;<=>
CDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~!"#$%&'()*+,-./0
56789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~!"
'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr
wxyz{|}~!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcd
ijklmnopqrstuvwxyz{|}~!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUV
[\]^_`abcdefghijklmnopqrstuvwxyz{|}~!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGH
MNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~!"#$%&'()*+,-./0123456789:
?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz
```

DIVERSE EXTRA P-CHARME STATEMENTS

Hier is dan een eerste set van extra statements voor gebruik met P-Charme. Als deze programma's stuk voor stuk worden gesaved met behulp van het BSAVE statement, dan kunnen ze op een simpele manier worden geassembleerd en in de tabel worden gezet. Voorbeeld:

```
!#6800=#80C6E3FF
T=#6803
A=#6880
*RUN "ATTACH"
*RUN "EXTERN"
*RUN "VLIST"
   enz.
?#3FC=#68
```

De statements zijn nu beschikbaar en worden herkent door de P-Charme interpretor. Rest nog een korte beschrijving van de diverse statements:

VLIST Vocabulary list **VL.**

Dit statement geeft een lijst van alle extra P-Charme statements die op dat moment door de interpreter herkend worden. Die lijst begint steeds met het pagina-nummer van de tabel met daaronder de statements en hun start-adres.

CLS Clear screen **CLS**

Met behulp hiervan kan het scherm worden schoongemaakt. Het voordeel van CLS boven PRINT $12 is dat CLS ook werkt als het scherm uit staat (met PRINT $21) en dat als de cursor uit staat dat ook zo blijft.

ATTACH Attach library **AT.**

Dit statement moet altijd gebruikt worden voordat de externe procedures of functies kunnen worden gedeclareerd. ATTACH moet worden gevolgd door een (correcte !) naam van een programma (zie pagina 12 P-Charme handleiding). Als dat programma niet in het geheugen staat volgt een foutmelding waarvan het nummer afhankelijk is van de plaats waar het ATTACH-statement ge-assembleerd werd. Na ATTACH kunnen alle procedures en functies uit het betreffende voorbeeld worden gebruikt als externe procedure (of functie). Zie voorbeeld programma verderop.
ATTACH mag binnen een programma meerdere keren worden gebruikt om procedures op te halen uit verschillende programma's.

PS. In een externe procedure of functie behoren geen GOTO's of GOSUBs te worden gebruikt. Deze springen namelijk naar het hoofdprogramma!!! (logisch). (kan eventueel worden opgelost door tijdelijk(!) de waarde van ?18 te wijzigen).

EXTERN EX.

Dit statement mag pas gebruikt worden nadat met behulp van ATTACH een andere programma is aangewezen om te worden gebruikt als bibliotheek van procedures en functies. Uit zo'n bibliotheek (zie voorbeeld STRING.LIB) kunnen met behulp van EXTERN de benodigde procedures worden geselecteerd. EXTERN moet worden gevolgd door het woord PROC of FUNCTION dat dan niet afgekort mag zijn, met daarachter de naam van de procedure (of functie) die men uit die bibliotheek wil gebruiken. In de bibliotheek zelf moet PROC of FUNCTION ook het eerste statement van een regel zijn en mag het niet worden afgekort. Zie ook het programma VOORBEELD.

\# #

Dit zeer korte statement moet direct gevolgd worden door een hexadecimaal getal en komt in de plaats van:

```
?18=..;NEW
    of
?18=..;END
```

afhankelijk van of er wel of geen OD staat aan het begin van de textpage. Voorbeeld:

```
#29
```

DIS Dis-assemble DIS

Dit statement begint van een opgegeven adres te dis-assembleren. In feite is dit de Miracle dis-assembler uit Acorn Nieuws van 1983. Binnenkort zal er echter een nieuw DIS statement worden gepubliceerd dat dan is uitgebreid met een zeer fraai STEP statement dat zeer veel mogelijkheden biedt (hoop ik). Voorbeeld:

```
DIS #A002
```

[P] [P]

Dit korte statement - een ge-inverteerde p - is bedoeld om in direct-mode (vanaf het toetsenbord) te worden gebruikt. Het verzet de BRK-vector naar P-Charme zodat bv. multidimensionale array kunnen worden afgedrukt. Het gebruik van [P] is echter aan enig risico verbonden. Voorbeeld:

```
[P]PRINT SS(1,12)
```

Merk op dat dit statement niet noodzakelijkerwijs gevolgd moet worden door een ';'. Het gebruik ervan binnen een programma is af te raden.

```
 10 PROGRAM VLIST.STAT
 20
 30 DIM LL5;F.I=0TO5;LLI=999;N.
 40 PRINT $21;P=A;GOSUB a
 50 PRINT $6 ;P=A;GOSUB a
 60 $T="VLIST";T=T+LEN(T)
 70 ?T=LL0/2560#80;T?1=LL0%256;T?2=#80;T=T+2
 80 A=P;T!1=A
 90 END
100
110a[
120:LL0;JSR #C4E4;LDA #3FC;STA #95;
130:LL1;LDY @0;STY #93;JSR #CD54
140LDA #95;STA #94;LDA (#93),Y;STA #95
150INY;LDA (#93),Y;CMP @#E3;BNE LL4
160INY;ASL A;CMP (#93),Y;BNE LL4
170LDA #94;JSR #F37E;LDA @CH":";JSR #CA4C;JSR #CD54
180:LL2;LDX #7;CPX @30;BNE LL3;JSR #CD54
190:LL3;INY;LDA (#93),Y;CMP @#80;BEQ LL1
200BPL LL5;JSR #CA4C;BPL LL3
210:LL4;JMP #C55B
220:LL5;JSR #F379;LDA #7;AND @#0F;CMP @#0A;BNE LL5
230LDA #94;ORA @#7F;AND (#93),Y;JSR #F37E
240INY;LDA (#93),Y;JSR #F376;JSR #F379;JMP LL2
250]
260 RETURN
```

```
 10 PROGRAM CLS.STAT
 20
 30 DIM LL0
 40 P=A;GOSUB a
 50 $T="CLS";T=T+LEN(T)
 60 ?T=LL0/2560#80;T?1=LL0%256;T?2=#80;T=T+2
 70 A=P;T!1=A
 80 END
 90
100a[:LL0;JSR #C4E4
105JSR #FD6D;JMP #C55B
110]
120 RETURN
```

```
 10 PROGRAM p.STAT
 20
 30 DIM LL0
 40 P=A;GOSUB a
 50 $T="p";T=T+LEN(T)
 60 ?T=LL0/2560#80;T?1=LL0%256;T?2=#80;T=T+2
 70 A=P;T!1=A
 80 END
 90
100a[
110:LL0;JSR #A42E;JMP #C31B
140]
150 RETURN
```

```
 10 PROGRAM ATTACH.STAT
 20
 30 DIM LL3;F.I=0TO3;LLI=999;N.
 40 PRINT $21;P=A;GOSUB a
 50 PRINT $6 ;P=A;GOSUB a
 60 $T="ATTACH";T=T+LEN(T)
 70 ?T=LL0/256@#80;T?1=LL0%256;T?2=#80;T=T+2
 80 A=P;T!1=A
 90 END
100
110a[
120:LL0;JSR #A60F;LDA @#0F;STA #32
130:LL1;INC #32;LDA #32;CMP @#A0;BEQ LL3+1;STA #92
140:LL3;LDA @0;STA #91;LDY @3;
150:LL2;INY;LDA (#91),Y;CMP #A3AD,Y;BEQ LL2;CPY @#0B;BNE LL1
160JSR #A57E;TYA;LDY #92;TAX;STX #93;INX;JSR #A544
170BCS LL1;JMP #C558
190]
200 RETURN
```

```
 10 PROGRAM EXTERN.STAT
 20
 30 DIM LL12;F.I=0TO12;LLI=999;N.
 40
 50 PRINT $21;P=A;GOSUB a
 60 PRINT $6 ;P=A;GOSUB a
 70 $T="EXTERN";T=T+LEN(T)
 80 ?T=LL0/256@#80;T?1=LL0%256;T?2=#80;T=T+2
 90 A=P;T!1=A
100 END
110
120a[
130:LL0;JSR #A60F;LDX @#FF;
140:LL1;STX #56;DEY;
150:LL2;INX;LDA #A3A1,X;BMI LL4;INY;CMP (#05),Y;BEQ LL2;
160:LL3;INX;LDA #A3A1,X;BPL LL3
170INX;LDY #03;CPX @#0A;BCC LL1;BRK
180:LL4;INY;STY #3;JSR #A60F;STY #95
190LDY @0;STY #58;LDA #32;STA #59
200:LL5;JSR #A277;TYA;CLC;ADC #58;STA #58;BCC LL6;INC #59
210:LL6;LDY @1;LDA (#58),Y;BMI LL4+9;INY
220:LL7;INY;LDA (#58),Y;CMP @#20;BEQ LL7;DEY;LDX #56
230:LL8;INX;LDA #A3A1,X;BMI LL9
240INY;CMP (#58),Y;BEQ LL8;BNE LL5
250:LL9;LDA #58;STA #91;LDA #59;STA #92;JSR #A57D
260TYA;CLC;ADC #91;BCC LL10;INC #92
270:LL10;PHA;TAX;LDY #92;TYA;PHA;TXA;STX #93;INX;JSR #A544
280LDY #94;DEY;BCS LL5
290:LL11;LDA #03;PHA;LDY #95;STY #03;JSR #A52B;BNE LL12
300PLA;STA #03;PLA;STA #28BD,X;PLA;STA #28DE,X;JMP #C558;
310:LL12;JMP #A5A1
320]
330 RETURN
```

```
  10 PROGRAM DIS.STAT
  20 DIM LL(20),TT(1)
  30 FOR I=0 TO 20;LL(I)=999;NEXT
  40 TT(0)=999;TT(1)=999
  50 V=#70
  60 PRINT $21;P=A;GOSUB a
  70 PRINT $6 ;P=A;GOSUB a
  80 $T="DIS";T=T+LEN(T)
  90 ?T=LL0/256@#80;T?1=LL0%256;T?2=#80;T=T+2
 100 A=P;T!1=A
 110 END
 120
 130a[
 140:LL0;JSR #C4E1;LDY @V;LDX #4;JSR #C3CD;JSR LL20;JMP #C55B;
 150:LL20;JSR LL1;BNE LL20;
 160:LL1;LDY @#2;
 170:LL2;LDA (V),Y;STA #66,Y;DEY;BPL LL2;
 180:LL3;INY;STY #F;LDX @#40;
 190:LL4;LDA #66;CMP @#D4;BEQ LL6;SEC;SBC #F1F3,Y;SEC
 200SBC #F210,X;BNE LL6;STA #65;LDY #F250,X;
 210:LL5;ROR A;ROR #65;DEY;BNE LL5;LDY #F;AND #F1D5,Y
 220BNE LL7;LDA #F1E4,Y;AND #65;BNE LL7;
 230:LL6;DEX;BNE LL4;CPY @#E;BNE LL3;LDY @#0;LDX @#11;STY #F;
 240:LL7;LDA #F202,Y;CMP @#F;BNE LL8;LDA @#1;
 250:LL8;AND @#F;STA #69;STA #0;STX #6A;LDX @V;JSR #F7F1
 260LDX @V;LDY @#0;
 270:LL9;LDA (V),Y;JSR #F7FA;JSR #FA08;BNE LL10;PLA;PLA;
 280:LL10;DEC #69;BPL LL9
 290LDY #E0;JSR #F99A;LDX #6A;LDA #F194,X;STA #64;LDA #F154,X
 300STA #65;LDY @#3;
 310:LL11;LDX @#5;
 320:LL12;ROL #64;ROL #65;ROL A;DEX;BNE LL12;AND @#1F;CLC
 330ADC @#3F;JSR #FFF4;DEY;BNE LL11
 340JSR #F7FD;LDY @#8;LDX #F;LDA TT0,X;STA #69;
 350:LL13;CPY @#6;BNE LL18;TXA;BNE LL15
 360LDA #67;BPL LL14;DEX;
 370:LL14;CLC;ADC V;STA #67;TXA;ADC V+1;STA #68
 380LDX @#2;BNE LL16;
 390:LL15;LDX #0;BEQ LL18;
 400:LL16;LDA @#23;JSR #FFF4;
 410:LL17;LDA #66,X;JSR #F802;DEX;BNE LL17;
 420:LL18;ASL #69;BCS LL19;LDA TT1,Y;JSR #FFE9;
 430:LL19;DEX;DEY;BPL LL13;JMP #C504;
 440]
 450 TT(0)=P;S=P
 460 !P=#F9FFFEFF;P!4=#7F7FFFCF;P!8=#87B1CFF9;P!12=#F9B7FF
 470 P=P+15
 480 TT(1)=P
 490 !P=#0D;P!1=#292C5941;P!5=#40282C58;P=P+9
 500 RETURN
```

```
 10 PROGRAM #.STAT
 20
 30 DIM LL0
 40 P=A;GOSUB a
 50 $T="#";T=T+LEN(T)
 60 ?T=LL0/256O#80;T?1=LL0%256;T?2=#80;T=T+2
 70 A=P;T!1=A
 80 END
 90
100a[
110:LL0;JSR #C90A;JSR #C4E4
120DEY;LDA #52;STA #12;STA #55
130LDA (#54),Y;CMP @#0D;BNE P+5;JMP #CD9B;JMP #C2B6
140]
150 RETURN
```

```
 10 PROGRAM VOORBEELD
 20
 30 ATTACH STRING
 40 EXTERN FUNCTION LEFT
 50 EXTERN FUNCTION RIGHT
 60 EXTERN FUNCTION MID
 70
 80 DIM S64
 90 $S="THIS IS A STRING"
100 PRINT $LEFT(S,6)'
110 PRINT $RIGHT(S,10)'
120 PRINT $MID(S,2,6)'
130 END
```

```
 10 PROGRAM STRING.LIB
 20
 30 FUNCTION RIGHT(S,L)
 40  $#140=$S;IF L<LEN(S) $#140=$S+LEN(S)-L
 50  RIGHT=#140
 60 FEND
 70
 80 FUNCTION LEFT(S,L)
 90  $#140=$S;L?#141=13
100  LEFT=#140
110 FEND
120
130 FUNCTION MID(S,L,M)
140  ?#140=13;IF L<=LEN(S) $#140=$S+L-1;M?#140=13
150  MID=#140
160 FEND
170
180 END
190
200      ZIE OOK PAGINA 31 P-CHARME HANDLEIDING
```

```
*****************************************************************
          J O Y S T I C K   A A N P A S S I N G E N
*****************************************************************

RATTRAP: ?#3408=#00 ?#3410=#02 ?#3425=#00 ?#342D=#08
         ?#3441=#01 ?#3449=#02 ?#345C=#01 ?#345F=#08
         REGEL NR.190 ?#B000=#F0;IF ?#B001&1=0 Q?2=0
         REGEL NR.191 ?#B000=#F1;IF ?#B001&1=0 Q?2=0

ACORN KONG:
         REGEL NR.3030...........Q=19............
         REGEL NR.3040...........Q=39............
         REGEL NR.3050...........Q=29............
         REGEL NR.3060...........Q=49............
         REGEL NR.3080.......?#B001%2=1 OR Q=29..........

BABIES:
         REGEL NR.10005 IF?#B001&2=0..............
         REGEL NR.10010 IF?#B001&8=0..............

WARLORDS: ?#3509=#F0 ?#3511=#10 ?#3517=#F0 ?#351F=#04
          ?#3527=#F1 ?#352F=#10 ?#3535=#01 ?#353B=#04

          ER ZIJN TWEE JOYSTICKS NODIG OM TE SPELEN!

OMEGA MISSION: ?#30DF=#10 ?#30FF=#04 ?#309E=#27
               ?#30BE=#13 ?#3321=#01 ?#3324=#01

STOCKCAR: !#3002=#D808F1F0 !#3006=#0202F1F0 ?#30F6=9
          !#300A=#1010F1F0 !#300E=#0404F1F0

          OOK HIER ZIJN TWEE JOYSTICKS NODIG!

INVADERS (VAN BUG-BYTE): ?#330F=#F0 ?#3317=#02 ?#3321=#08
               ?#3364=#4E ?#3365=#01 ?#3366=#B0 ?#3367=#90

ASTEROIDS: ?#2EB1=#14 ?#2EB2=#00 ?#2EB3=#3B ?#2EB4=#29
           ?#2EB5=#10 !#2E95=#D03B1420 ?#2E99=#03
           !#2EA5=#D03B1F20 ?#2EA9=#03

P=#3B00;[LDA@#F0;STA#B000;LDA#B001;RTS;LDA@#F0;STA#B000
[LDA#B001;AND@2;RTS;LDA@#F0;STA#B000;LDA#B001;AND@8;RTS
[LDA@#F0;STA#B000;LDA#B001;AND@1;RTS

PAINTER:   ?#2E4B=#90 ?#2E53=#01 ?#2E5A=#90 ?#2E62=#10
           ?#2E69=#90 ?#2E71=#04 ?#2E78=#90 ?#2E80=#02
           ?#2E84=#90

FROGGER: ?#36A7=#27 ?#36AB=#19 ?#36AF=#13 ?#36B3=#31

TORPRUN: ?#3192=#D0 ?#3264=#D0 ?#326C=#02
```

```
ATTACK:    ?#29E6=#00 ?#29EE=#02 ?#2A07=#00 ?#2A0F=#08
           ?#2A28=#00 ?#2A30=#10 ?#2A3C=#00 ?#2A44=#04
           ?#2B9B=#00

SWARM: ?#3106=#08 ?#30F4=#01 ?#30F7=#01 ?#311A=#13

HYPER: ?#35BA=#08

P=#35C9;[JSR#3BE8;LDA#B001;AND@#10;BNE#35DD;DEC#70;LDA#70
[CMP@#10;BNE#35EC;BEQ#35EA;LDA#B001;AND@4
[BNE#35EC;LDY#70;CPY@#BB;BEQ#35EC
P=#3BE8;[LDA@#D0;STA#B000;RTS;LDA#B001;AND@1;RTS
P=#3557;[JSR#3BEE;BNE#35B6

 3D-ASTEROIDS:
   REGEL NR.130 ?R=#70;IF?S=253.......................
   REGEL NR.140 ?R=#70;IF?S=247.......................
   REGEL NR.150 ?R=#70;IF?S=251.......................
   REGEL NR.160 ?R=#70;IF?S=239.......................

SNOOKER: !##3626=#F0F0F0F0 !#362A=#10040802
         REGEL NR.6520 T=?#B001&1
```

****

Tot slot nog een opmerking over de snelheid van sommige spelen. Meestal wordt hiervoor (in machinetaal) gelinkt naar een subroutine op #FE66.
In deze subroutine wordt de schrijfstraal gesynchroniseerd waardoor er geschreven wordt zonder storing in beeld.
Door deze link naar #FE66 te verwijderen wordt een spel meestal veel sneller, NIET ALTIJD!!.
Dit gaat wel gepaart met ruis, wat soms hinderlijk is.
Hieronder staat een programmatje wat deze link verwijdert:

```
 10 FOR I=#2900 TO #3C00

 20 IF ?I=#20 AND I?1=#66 AND I?2=#FE GOSUB.a

 30 NEXT I

 40 END

 50a PRINT &I'

 60 ?I=#EA; I?1=#EA; I?2=#EA

 70 RETURN
```

Alle links naar #FE66 worden nu omgezet in "NOP", als het nu te snel wordt kun je voor enkele van de uitgeprintte adressen (uit bovenstaand programma) invoeren:
P=#ADRES;[JSR#FE66
Met een beetje experimenteren kom je er wel uit.

SUCCES!!

Voor een aantal spelletjes zijn in AN reeds aanpassingen beschreven voor een joy-stick aansluiting .Echter deze aanpassingen werken alleen als slechts 1 joystick aangesloten is;d.w.z. de gemeenschappelijke leiding ligt aan massa en niet aan pen 1 van IC 26 .Daarom dit artikeltje voor de mensen die hun joystick in de 2 sticks versie aangesloten hebben .Ik bespreek natuurlijk enkel de programma's die ik zelf heb . Naast een aantal programma's uit de oorspronkelijke artikels van Jan Wijnen in AN zal ik ook nog een reeks van andere spelletjes bespreken,die ik zelf omgebouwd heb.

ACORNSOFT,SNAPPER:werkt goed zoals beschreven in AN 1,'83 blz.51

ACORNSOFT,INVADERS:werkt goed zoals beschreven in AN 2,'83 blz.39

ACORNSOFT,BREAKOUT&SPACE ATTACK:zie AN 3,'83 blz.51

ASTROBIRDS:dit is een andere versie dan die uit AN 3 '83 blz.51 . Oorsprong is mij onbekend. Aanpassingen:

?≠31DE=8 ?≠31F1=0 ?≠31F9=2 ?≠3256=1 ?≠3259=1

A&F SOFTWARE,CYLONATTACK:

?≠3369=16 ?≠3377=4 ?≠3389=8 ?≠339B=2 ?≠334E=≠F0
?≠3361=≠F0 ?≠336F=≠F0 ?≠3381=≠F0 ?≠3393=≠F0

ACORNSOFT,DODGEMS:

?≠3369=10 ?≠336B=10 ?≠3374=2 ?≠3496=1 ?≠3499=1

MARTIANS:Oorsprong onbekend;

?≠3813=2 ?≠380B=0 ?≠3817=0

A&F SOFTWARE,PAINTER:

?≠2E4B=≠90 ?≠2E53=1 ?≠2E5A=≠90 ?≠2E62=≠10 ?≠2E69=≠90 ?≠2E71=4
?≠2E78=≠90 ?≠2E80=2 ?≠2E84=≠90 ?≠2E8C=8

MICROMANIA,OMEGA MISSION: het stuk DATA hoeft niet gewijzigd.Voor het eerste stuk: ?≠30DF=≠10 ?≠30FF=4 ?≠3184=1 ?≠3187=1 ?≠3196=1 ?≠3199=1
?≠3321=1 ?≠3324=1 ?≠3347=1 ?≠309E=≠27 ?≠30BE=≠13

WARLORDS:oorsprog onbekend,

?≠3509=≠F0 ?≠3511=≠10 ?≠3517= ≠F0 ?≠351F=4 ?≠3527=≠F1 ?≠352F=≠10
?≠3538=4

STOCKCAR:oorsprong onbekend,

?≠3002=≠F0 ?≠3003=≠F1 ?≠3004=8 ?≠3005=8 ?≠3006=≠F0 ?≠3007=≠F1
?≠3008=2 ?≠3009=2 ?≠300A=≠F0 ?≠300B=≠F1 ?≠300C=≠10 ?≠300D=≠10
?≠300E=≠F0 ?≠300F=≠F1 ?≠3010=4 ?≠3011=4

MICROMANIA,PUCKMAN:

?≠3217=29 ?≠3228=49 ?≠3239=19 ?≠324A=39

RK CEMENT:

```
130IF?#84=19;X=X-1
140IF?#84=39;X=X+1
180IF?#84=9;GOTOg
```

SHOOTOUT(COWBOY):oorsprong onbekend.

```
?#380F=#F0 ?#3817=#10 ?#381B=#F0 ?#3823=4   ?#3827=#F0 ?#382F=1
?#3833=#F1 ?#383B=#10 ?#383F=1   ?#3842=4   ?#3846=#F1 ?#384E=1
```

FROG JUMP (FROGGER):oorsprong onbekend.

```
?#36A7=39  ?#36AB=29  ?#36AF=19  ?#36B3=49
```

ACORNSOFT,SNOOKER:

```
?#3626=#F0 ?#3627=#F0 ?#3628=#F0 ?#3629=#F0 ?#362A=2  ?#362B=8
?#362C=4   ?#362D=#10
6520T=?#B001&1
```

ACORNSOFT,ASTROIDS:

```
?#2E01=#F0
```

Ga dan naar tekstspace #82 en type volgend programma in:

```
 1ØDIMLL4;F.I=ØTO4;LLI=#FFFF;N.
 2ØF.I=1TO2
 3ØP=#2E8D;[
 4ØLDA#BØØ1;LSRA;STA#7F;BCSLL1;JSR#315A
 5Ø:LL1 LSR#7F;BCSLL2;DEC#39AØ
 6Ø:LL2 LSR#7F;LSR#7F;BCSLL3;INC#39AØ
 7Ø:LL3 LDA#39AØ;AND@#7F;STA#39AØ;LDA@Ø;STA#95;LSR#7F;BCSLL4
 8ØLDA@#FF;STA#95;LDX@Ø
 9Ø:LL4 JSR#3234
1ØØ]
11ØN.;E.
```

Laat dit programma runnen . Nu is ASTROIDS klaar voor gebruik met joystick . Het zojuist ingetypte programma is nu niet meer nodig.

Alle vermelde veranderingen zijn bij mij in gebruik en werken perfect.Indien U de veranderingen zorgvuldig overneemt (onder de voorwaarde dat ik ze hopelijk zorgvuldig heb neergeschreven) dan zouden ze ook bij U moeten werken.

ATOM POWER PLOTTING

Onlangs kreeg ik uit ons drukwerkarchief het artikel"ATOM PLOTTING POWER" in handen.De grafische mogelijkheden die door dit programma gecreëerd worden leken mij zo veelzijdig dat ik het niet kon laten om hierover een artikel te schrijven opdat iedereen in de club zou weten welke de mogelijkheden zijn die dit programma biedt.
Het geassembleerde programma kan vrij eenvoudig in een basic programma gebruikt worden.Het programma kan individuele shapes of groepen van shapes plotten,wissen of verplaatsen en dit in elke grafische mode.Door gebruik te maken van de kleurencommando's kunnen de shapes in verschillende kleuren of tinten geplot worden.
Het programma maakt gebruik van array's waarin de elementen van een shape en de eigenschappen ervan worden opgeslagen.Men hoeft dus maar juist de elementen van een bepaalde array te wijzigen om uw eigen shapes te creëren.
Het programma maakt gebruik van volgende array's.

XX = de x-coördinaat op het scherm.

YY = de y-coördinaat op het scherm.

AA = de eigenschappen van een bepaalde shape.

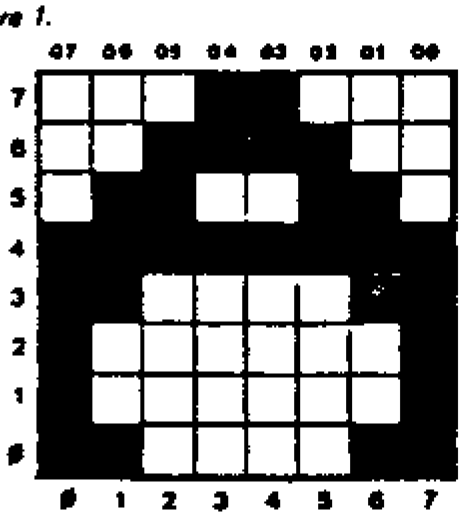
DD = bevat de plotpunten van een shape.

U kunt maximaal 256 shapes tegelijkertijd gebruiken.Hiervoor hebt U ± 5K ram nodig. In de praktijk volstaat men met 10 à 20 shapes.Het aantal shapes dat men wilt gebruiken moeten eerst benoemd worden.
Voor één shape is de benoeming de volgende.

```
10 DIM DD(1),AA(0),XX(0),YY(0)
```

Nu onze shape tabel. Deze bestaat uit 64 plotpunten;gerangschikt in acht rijen van 8 bits. Veronderstel nu even dat we de tekening van figuur 1 willen opslagen in on shapetabel.Dit is een typisch voorbeeld van een invader. U ziet duidelijk de opbouw van 8 X 8 bits. Of beter gezegd 8 bytes.

Figure 1.



De donkere velden worden binair 1 de andere velden worden nul. Nadien zetten we deze om in Hexadecimaal.Voor onze invader wordt dit:

BINAIR:	HEX.
00011000	18
00111100	3C
01100110	66
11111111	FF
11000011	C3
10000001	81
10000001	81
11000011	C3

Deze tekening slaan we op in array DD en wel op volgende manier.

```
20 DD(0) = #FF663C18
30 DD(1) = #C38181C3
```

Denk eraan dat de elementen van een array omgekeerd worden opgeslagen in een array.De plaats op het scherm wordt bepaald door de waarden opgeslagen in de array's XX en YY. Deze waarden zijn absolute waarden.Dus altijd gerekend vanaf het punt van oorsprong links onderaan.

```
40 XX(0)=0;YY(0)=0
```

Deze plaats is dus de plaatsbepaling van het links-beneden element van uw shape.

Zover zijn we nu.Nu is natúurlijk de vraag. Wat kunnen we hiernu allemaal mee doen? Wel heel wat.
Wat we doen wordt bepaald door de inhoud van het overeenkomstige element in array AA. Ieder element bestaat uit 4 bytes.Iedere byte heeft zijn eigen specifieke funktie. Ter verduidelijking worden de bytes als volgt benoemd.

msb | | | lsb

MAGNIFY	CHAIN SHAPE	PIXEL BLOCK	STATUS
M	C	N	S

Eérste byte M: of de Most Significant Byte.Deze bepaalt de afmetingen van uw shape op het scherm.De eerste vier bits bepalen de vergrotingsfactor in de x-richting.De laatste 4 bits bepalen de vergroting in de y-richting.VB. M=O De shape wordt geplot zonder enige vergroting;d.w.z. elke bit van uw shape komt overeen met één punt op het scherm.
Nemen we M=#11 dan beslaat elke bit van uw shape 2 punten op het scherm.zowel in de x- als in de y-richting.Nemen we M= #FO dan beslaat elke bit 16 punten op het scherm in de x-richting en 1 punt in de y-richting.Het beeld wordt dus uitgerokken in de breedte.
Tweede byte C:Wordt gebruikt in combinatie met de chain-bit van de statusbyte en dit om shapes onderling te koppelen.Veronderstel even dat shape 1 gekoppeld is aan shape O.Telkens nu shape 1 geplot of gewist wordt of verplaatst; gebeurd dit eveneens met shape O.Dit is zeer gemakkelijk wanneer U tekeningen wilt gebruiken die niet in één shape kunnen geplaatst worden.Bijvoorbeeld een tekening van 18 X 8 punten.Door ze onderling te koppelen worden zij als één shape behandeld.De waarde van C is het shapenummer dat gekoppeld is en kan variëren van O tot #FF.

VB.: Shape 1 is onafhankelijk C= OO
Shape 1 is gekoppeld aan 2 C= O1
Shape 2 is gekoppeld aan 3 C= O2

Wordt nu shape 3 geplot dat zit het programma aan de inhoud van byte C in ons geval O2 dat shape 2 ook moet geplot worden.Bij het plotten van shape 2 ziet hij in de C-byte van shape 2 de waarde één staan zodat shape 1 ook geplot wordt.Er worden dus 3 shapes geplot door één commando.
Derde byte N:Bepaalt welke shape geplot moet worden.
Als byte N = OO dan wordt de tekening geplot welke zich in de elementen DD(o) en DD(1) bevindt.
Als byte N= O1 dan wordt DD(2) en DD(3) geplot
Als byte N= O2 dan wordt DD(4) en DD(5) geplot
Men kan dus door eenvoudig de waarde van byte N te veranderen van tekening veranderen.Alle andere bepalingen van de vorige shape blijven gehandhaafd.
Vierde byte S: of de STATUS byte. Hiervan heeft elke bit een afzonderlijke functie.

MSbit | | | | | | | LSbit

TRANS.	SNOW	CHAIN	FORE GROUND		SOUND	BACK GROUND	
1	1	O	O	1	O	O	O
D7	D6	D5	D4	D3	D2	D1	DO

We zullen nu iedere bit van het status-byte afzonderlijk behandelen.

D7: indien 1 :Dan worden de punten van uw shape die binair de waarde O hebben niet geplot.De bestaande achtergrond blijft dus gehandhaofd.Hij is transparant.

D6: indien 1 :dan worden de punten geset tijdens de flyback en zal er geen interferentie zijn op het scherm.Dit gaat dan wel ten koste van de snelheid.

D5: indien 1 :Dan worden de huidige uit te voeren plotopdrachten ook uitgevoerd op de shape die aangegeven wordt door byte C .

D4 en D3 bepalen de voorgrond. En wel op volgende wijze.

indien:		dan:	
O	O	clear	pixel
O	1	set	pixel
1	O	invert	pixel
1	1	clear	pixel

D2: indien 1 :dan wordt de luidspreker aangestuurd telkens een punt wordt geplot.

D1 en DO bepalen de achtergrond. En wel op volgende wijze.

indien:		dan:	
O	O	clear	pixel
O	1	set	pixel
1	O	invert	pixel
1	1	clear	pixel

N.B.: onder de voorgrond verstaan we die punten welke binair 1 zijn. De achtergrond zijn die punten welke binair O zijn.

Hebben we éénmaal uit deze eigenschappen gekozen.Dan dienen we ze op te slaan in array AA.Bijvoorbeeld. We kiezen voor een normale afmeting (M=O),ongekoppeld (C=OO),de te plotten shape bevindt zich in shape O (N=CO).De bestaande achtergrond moet blijven.(D7 =1),geen interferentie (D6=1),geen koppeling (D5 = O),de voorgrond moet geset worden (D4=O),(D3=1),geen geluid (D2=O),de achtergrond blijft clear.(D1=O),(DO =O).
De statusbyte heeft dus binair de volgende waarde.

S = 11001000 binair of hex. CB

Het eindresultaat wordt dus.

```
50 AA(O)=#000000CB
```

Nu hebben we nog de keuze van plotten.

OPERATIE	SOURCE	OBJ.	VARIABLES. A	X	Y
Plot 'n shape	M+O	#2920	shape n°		
wis 'n shape	M+3	#2923	shape n°		
move 'n shape	M+6	#2926	shape n°	x-richt.	y-richt.
plot pixels	M+9	#2929	blok n°	x-posit.	y-posit.

Zoals we reeds besproken hebben moeten we om een shape te plotten eerst de arrays AA,DD,XX en YY benoemen.De shape nummer plaasten we nu in de basic-variabele A en we linken naar #2920.

```
60   CLEAR 4
70   A=0; LINK #2920
```

Om een bepaalde shape te wissen in het belangrijk te onthouden dat dit enkel kan gebeuren indien bit D7 van het status-byte is geset.Dus transparant en indien de achtergrondsbit de eigenschap clear heeft meegekregen.Om nu 'n bepaalde tekening te wissen plaatsen we het shape nummer in de variabele A en linken naar #2923

```
80 A=0;LINK #2923
```

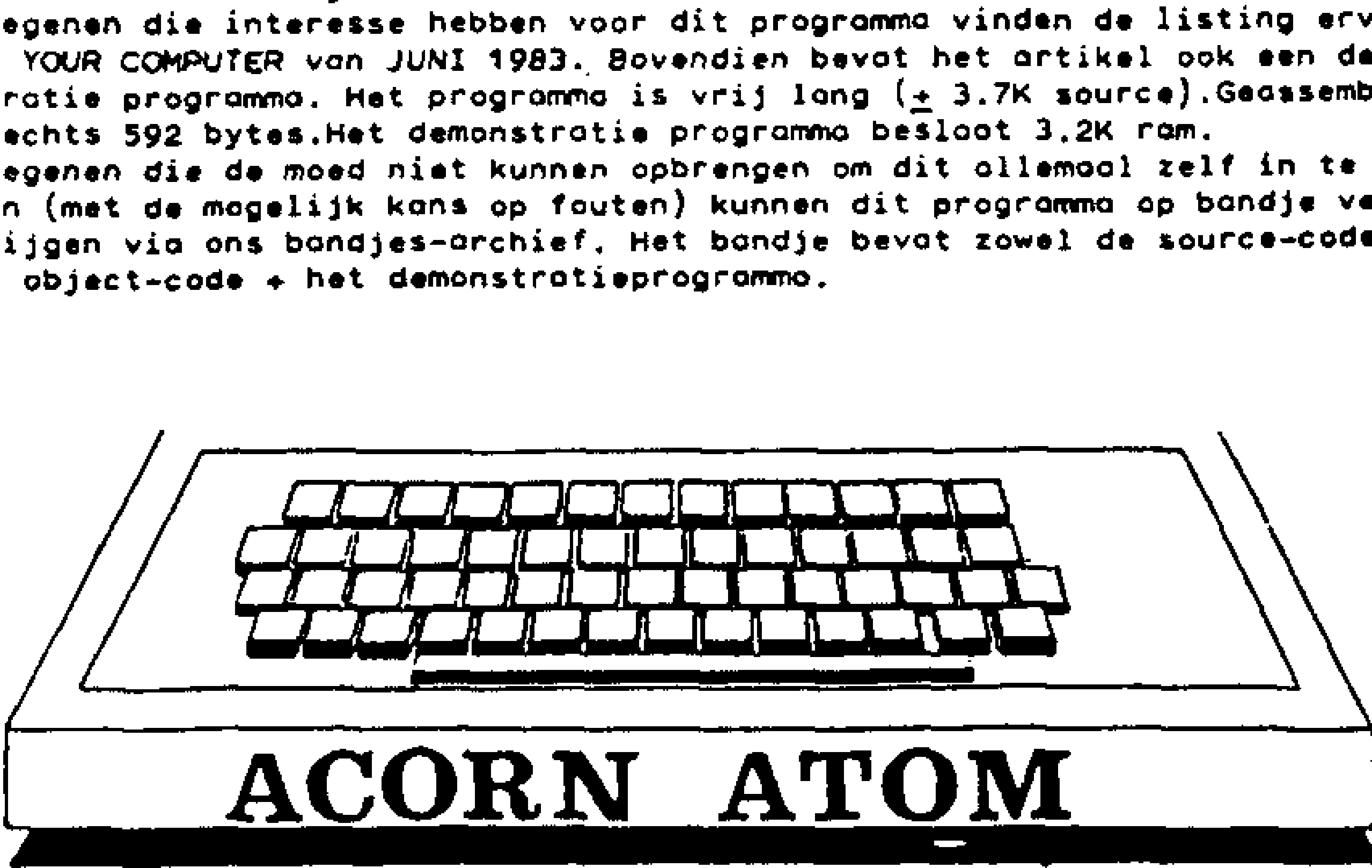
Nu de move-routine.Een geplotte shape kan verplaatst worden; of beter gezegd gewist en opnieuw geplot op een andere plaats.De nieuwe positie wordt bepaald door de waarde van de variabele X en de variabele Y.Deze waarden zijn verschilwaarden t.o.v. de uitgangspositie.
We plaatsen dus de shape nummer in de variabele A.De afstand die de tekening moet verplaatst worden op de X-as in de variabele X en de afstand van verplaatsing in de Y-richting in de variabele Y.Vervolgens linken we naar#2926.
B.V.: om shape 0 10 posities opwaarts en 20 posities naar rechts te verplaatsen geven we de volgende opdracht.

```
90 Y=10;X=20;A=0;link#2926
```

Merk op dat de waarde van X en Y zowel positief als negatief kunnen zijn en dat de overeenkomstige waarden in de XX en YY arrays worden aangepast.
Verder hebben we nog een mogelijkheid om een bepaalde shape te plotten zoals de plotpunten in werkelijkheid zijn; dus zonder enige optie.Dit gebeurd op een absolute plaats. Dus de plaats gerekend vanaf linksonderaan van het beeldscherm.We plaatsen het shapenummer in de variabele A, de horizontale positie in de variabele X en de verticale positie in de variabele Y en we linken naar #2929.

```
100 X=50;Y=50;A=0;LINK #2929
```

U ziet de mogelijkheden zijn enorm.Met behulp van deze routines een spelietje invaders schrijven in niet zo ingewikkeld meer.
Diegenen die interesse hebben voor dit programma vinden de listing ervan in *YOUR COMPUTER* van JUNI 1983. Bovendien bevat het artikel ook een demonstratie programma. Het programma is vrij lang (± 3.7K source).Geassembleerd slechts 592 bytes.Het demonstratie programma besloot 3.2K ram.
Diegenen die de moed niet kunnen opbrengen om dit allemaal zelf in te tikken (met de mogelijk kans op fouten) kunnen dit programma op bandje verkrijgen via ons bandjes-archief. Het bandje bevat zowel de source-code als de object-code + het demonstratieprogramma.



```
    10 PROGRAM GFILL.STAT
    20
    30 DIM FF39,RR3
    35 FOR I=0 TO 39;FFI=999;N.
    40 PRINT $21;P=A;GOSUB a
    50 PRINT $6 ;P=A;GOSUB a
    60 $T="GFILL";T=T+LEN(T)
    70 ?T=FF0/256@#80;T?1=FF0%256;T?2=#80;T=T+2
    80 A=P;T!1=A
    90 END
   100
   110 WERKT IN ALLE GRAFISCHE MODES, OOK IN DE 4-KLEUREN MODE
   120 ER KAN ALLEEN IN DE NIET-ACHTERGRONDKLEUR INGEVULD WORDEN
   130
   140aX=#57;Y=#52;B=#90;C=#91
   150 S=#92;V=#93;W=#94
   160 RR0=#95;RR1=#96;RR2=#99;RR3=#9A
   170[
   180:FF0 \gfill kommando
   190 JSR#C8BC;JSR#C231;JSR#C8BC;JSR #C4E4
   220 LDA#23,X;ORA#32,X;ORA#41,X;ORA#24,X;ORA#33,X;ORA#42,X
   230 BNE FF4
   240 LDA#14,X;STA X
   250 LDA#15,X;STA Y
   260 JSR FF1
   270:FF4
   280 DEC#04;DEC#04
   290 JMP#C55B
   300
   310:FF1 \entry fill
   320
   330 LDA@0;STA#5B;STA#5D;LDA X;STA#5A;LDA Y;STA#5C
   340
   350 LDA@2;STA#5E;JSR FF39 \bepaal adres door pixel te invert.
   360 LDY@0;LDA(#5F),Y;STA B \onthoud geinv.byte
   370 JSR FF39;LDY@0;LDA(#5F),Y;CMP B;BEQ FF2 klaar als buiten s
cerm
   380 STA B
   390 LDA@0;STA#5E;JSR FF39;LDA B;LDY@0;CMP(#5F),Y
   400 PHP;STA(#5F),Y;PLP;BNE FF2 klaar als pixel al geset
   410
   420 LDA@0;STA#5A;STA#5C;LDA@2;STA#5E;JSR FF39;JSR FF39 bepaal
current color
   430 LDY@0;LDA(#5F),Y;STA C
   440 STY#5E;JSR FF39 clear pixel
   450 LDY@0;LDA(#5F),Y;STA B
   460 LDA@1;STA#5E;JSR FF39 set pixel
   470 LDY@0;LDA(#5F),Y;CMP B;PHP;LDA C;STA(#5F),Y;PLP;BEQ FF2
   480 \klaar als current color = background color
   490
   500 LDA@0;STAS \stackpointer op 0
   510:FF20
   520 JSR FF5
   530 LDA#B001;BMI FF21 stop als shift ingedrukt
   540:FF2
   550 RTS
   560:FF21
```

```
 570 LDA RR2:BNE FF22
 580 INC X:JSR FF5:JMP FF21
 590:FF22
 600 JSR FF3 \kleur pixel
 610 LDA RR1:STA V:LDA RR3:STA W
 620 BNE FF23
 630 DEC Y:JSR FF31:INC Y
 640:FF23
 650 LDA RR1:BNE FF24
 660 INC Y:JSR FF31:DEC Y push x,y+1
 670:FF24
 680 LDA RR0:BNE FF25
 690 DEC X:JSR FF3
 700 JSR FF5
 710 LDA RR1:BEQ FF26
 720 LDA V:BNE FF26
 730 INC V
 740:FF26
 750 LDA RR3:BEQ FF27
 760 LDA W:BNE FF27
 770 INC W
 780:FF27
 790 LDA V:BEQ FF28
 800 LDA RR1:BNE FF28
 810 INC Y:JSR FF31 push x,y+1
 820 DEC Y:DEC V
 830:FF28
 840 LDA W:BEQ FF29
 850 LDA RR3:BNE FF29
 860 DEC Y:JSR FF31 push x,y-1
 870 INC Y:DEC W
 880:FF29
 890 JMP FF24
 900:FF25
 910 LDA S:BEQ FF30 klaar als stack leeg
 920 JSR FF32:JMP FF20 \pull X,Y
 930:FF30
 940 RTS
 950 \einde body
 960
 970:FF31 \push X,Y
 980 LDX S:LDA X:STA#0100,X:LDA Y:STA#0101,X:INX:INX:STX S
 990 CPX@#E0 max stackhoogte
1000 BCS FF33:RTS
1010:FF32 \pull X,Y
1020 LDX S:DEX:DEX:LDA#0101,X:STA Y:LDA#0100,X:STA X:STX S
1030 RTS
1040:FF33 PLA:PLA:JSR#FD1A:JSR#FD1A:RTS piep en stop want stack
vol
1050:FF36
1060
1070 \hulproutines voor inkleur routine
1080:FF3 \set pixel X,Y
1090 LDA@0:STA#5B:STA#5D:LDA  X:STA#5A:LDA  Y:STA#5C
1100 LDA@1:STA#5E:JMP(#3FE)
1110 \einde set pixel routine
1120
```

```
1130:FF5 \test 4 pixels rond X,Y
1140 LDA X;STA#5A;LDA@0;STA#5B;STA RR1
1150 LDA@1;CLC;ADC Y;STA#5C;LDA@0;ADC@0;STA#5D \boven
1160 LDA@2;STA#5E;JSR FF39 \bepaal adres
1170 LDY@0;LDA(#5F),Y;STA B \onthoud geinv.byte
1180 JSR FF39;LDY@0;LDA(#5F),Y;CMP B;BEQ FF16;STA B
1190 LDA@0;STA#5E;JSR FF39;LDA B;LDY@0;CMP(#5F),Y;BEQ FF6
1200 STA(#5F),Y
1210:FF16 LDA@1;STA RR1
1220:FF6
1230 LDA X;STA#5A;LDA@0;STA#5B;STA RR3
1240 LDA Y;SEC;SBC@1;STA#5C;LDA@0;SBC@0;STA#5D \onder
1250 LDA@2;STA#5E;JSR FF39 \bepaal adres
1260 LDY@0;LDA(#5F),Y;STA B \onthoud geinv.byte
1270 JSR FF39;LDY@0;LDA(#5F),Y;CMP B;BEQ FF17;STA B
1280 LDA@0;STA#5E;JSR FF39;LDA B;LDY@0;CMP(#5F),Y;BEQ FF7
1290 STA(#5F),Y
1300:FF17 LDA@1;STA RR3
1310:FF7
1320 LDA Y;STA#5C;LDA@0;STA#5D;STA RR0
1330 LDA  X;SEC;SBC@1;STA#5A;LDA@0;SBC@0;STA#5B \links
1340 LDA@2;STA#5E;JSR FF39 \bepaal adres
1350 LDY@0;LDA(#5F),Y;STA B \onthoud geinv.byte
1360 JSR FF39;LDY@0;LDA(#5F),Y;CMP B;BEQ FF18;STA B
1370 LDA@0;STA#5E;JSR FF39;LDA B;LDY@0;CMP(#5F),Y;BEQ FF8
1380 STA(#5F),Y
1390:FF18 LDA@1;STA RR0
1400:FF8
1410 LDA Y;STA#5C;LDA@0;STA#5D;STA RR2
1420 LDA  X;CLC;ADC@1;STA#5A;LDA@0;ADC@0;STA#5B \rechts
1430 LDA@2;STA#5E;JSR FF39 \bepaal adres
1440 LDY@0;LDA(#5F),Y;STA B \onthoud geinv.byte
1450 JSR FF39;LDY@0;LDA(#5F),Y;CMP B;BEQ FF19;STA B
1460 LDA@0;STA#5E;JSR FF39;LDA B;LDY@0;CMP(#5F),Y;BEQ FF9
1470 STA(#5F),Y
1480:FF19 LDA@1;STA RR2
1490:FF9
1500 RTS
1510 \einde test 4 pixels rond X,Y-routine
1520
1530:FF39 \plot pixel
1540 JMP(#3FE)
1550]
1560 RETURN
1570
```

```
 10 PROGRAM DEMO GFILL
 20
 30 CLEAR 3
 40 COLOUR 2
 50 MOVE 20,10
 60 DRAW 20,90;DRAW 100,90
 70 DRAW 100,10;DRAW 20,10
 80 MOVE 40,10
 90 DRAW 40,80;DRAW 80,80
100 DRAW 80,20;DRAW 60,20
110 DRAW 60,30;DRAW 65,30
120 DRAW 65,25;DRAW 60,25
130 COLOUR 1
140 GFILL 41,30;REM STARTPOSITIE INKLEUREN
150 END
```

SYMBOLISCHE ASSEMBLER

Een symbolische assembler is een assembler waarbij de labels en andere variabelen mogen bestaan uit een naam van 1 of meer letters en/of cijfers te beginnen met een letter. Het is duidelijk dat zo'n assemblerprogramma heel betekenisvol kan zijn: ENTRY, LOOP, PORTB, CRLF, TEXTPTR zeggen meer dan LLO, P-8, #B001, #FFED, #12. Het gemis van deze faciliteit in de Atom-assembler lijkt klein, maar is dat bij langere, ingewikkelde programma's met veel interactie zeker niet. Een toolkit (losse routines) kan nog heel goed met de Atom- assembler aangemaakt worden, maar bij bv. een database of intensief gebruik van I/O wordt het al heel snel behelpen. Een symbolische assembler is dus gewenst, maar hoe komen we eraan?

De makkelijkste manier (principieel dan!) is het uitbouwen van de Atom-assembler, waarbij je meteen nog meer faciliteiten kunt toevoegen. Op dit moment (inmiddels 4 maanden later) is een dergelijke assembler verwezenlijkt. Daarbij is sterk gekeken naar de MTAL-standaard (MTAL=MOS Technology Assembly Language, zie verder 'Programming the 6502', Rodnay Zaks) in combinatie met de Acorn filosofie.

Een greep uit de wijzigingen:

1. Labels bestaan uit 1 of meer letters en/of cijfers te beginnen met een letter en voorafgegaan door een dubbele punt (:)
   - NB1 'A' mag niet als label gebruikt worden; is voorbehouden aan de accu
   - NB2 een letter is een hoofd- of kleine letter
   - NB3 de 'single quote' (') is toegestaan t.b.v. een betere leesbaarheid bv. NEXT'CHAR i.p.v. NEXTCHAR
   - NB4 de labellengte is technisch 'beperkt' tot ongeveer 250 karakters. De buffer staat op #288D, zodat 64 zonder problemen gebruikt kunnen worden. Wie hier niet genoeg aan heeft mag het zeggen.
   - NB5 een te lange labellengte genereert geen foutmelding (in praktijk niet relevant)
2. Binaire constanten zijn toegestaan, voorafgegaan door de procent (%) bv. LDA @%1010 0101 is hetzelfde als LDA@#A5 (spaties zijn facultatief, leidende nullen ook)
3. Er zijn 'directives' toegevoegd

```
.BYTE V          i.p.v. ];?P=V;P=P+1;[
.WORD V          i.p.v. ];!P=V;P=P+2;[
.DBYTE V         i.p.v. ];?P=V/256;P?1=V%256;P=P+2;[
.ASCII/TEKST/    i.p.v. ];$P="TEKST";P=P+LEN(P);[
.PAGE            i.p.v. ];PRINT $12;[
.SKIP N          i.p.v. ];FOR I=1 TO N;PRINT ';NEXT I;[
.END             i.p.v. ];NEXT I;PRINT $6;END   o.i.d.
.SCRON en .SCROFF regelen het scherm
.PRON  en .PROFF  regelen de printer
```

4. Karakter expressies
   CH is vervangen door ' bv. LDA @'> i.p.v. LDA @CH">"
5. Ster expressies
   de 'object code pointer' wordt bepaald en gelezen via *
   bv. *=#2000, BNE *-4

6. High en low byte specificaties
   > geeft het high byte van de expressie, < het low byte
   bv. .ADRES=#1234
```
        LDA @ADRES>         is hetzelfde als LDA @#12
        LDA @ADRES+1<       is hetzelfde als LDA @#35
        LDA @ADRES>+1       is hetzelfde als LDA @#13
                              of als LDA @ADRES+#100>
```
7. Out of range detectie
   Aan het eind van de vertaling wordt er een melding gegeven als er een 'out of range' heeft plaatsgevonden (ook bij .SCROFF)
8. Er wordt een waarschuwing gegeven als een label of variabele niet gedeclareerd is of als deze meerdere malen wordt gedefinieerd
9. Declaratie van een variabele gebeurt met de punt (.)
   bv. .SCREEN=#8000
10. Er worden 'echte' foutboodschappen gegeven
11. Er wordt een melding gegeven als een 'branch' naar een andere pagina plaatsvindt, i.v.m. de extra clock cycle
12. Normaliter zijn het scherm en de printer tijdens pass 1 uit, het scherm aan tijdens pass 2. Andere instellingen kunnen met de directives verwezenlijkt worden. Tijdens pass 1 is de printer altijd uit.

De hier beschreven assembler zal als RELEASE 0.0 worden vrijgegeven vanaf 25/26 november. Deze versie is in de praktijk getest (dat klinkt leuk, maar betekent eigenlijk dat hij niet 100% foolproof is, maar slechts ongeveer toch zeker meer dan laten we zeggen iets in de buurt van pak 'm beet 98%). Eventuele fouten zullen zo mogelijk in RELEASE 0.1 worden verbeterd. Ondertussen wordt de basis gelegd voor RELEASE 1.0 met: .OPTION en .TITLE, mogelijkheid tot opnemen in een Basic programma (zoals we dat nu gewend zijn), gesorteerde symbol table en een faciliteit om de object code elders op te slaan.

Uitleg van de symbolische assembler.

De symbolische assembler lijkt sterk op de Atom-assembler in die zin dat de syntax hetzelfde is gebleven. Eén faciliteit moet de huidige versie vooralsnog missen: de mengbaarheid met Basic. Dit houdt o.m.in dat macro's en "conditional assembly" ook niet meer mogelijk zijn. We krijgen er echter iets belangrijks voor terug: de mogelijkheid om getallen en labels te representeren door een zgn. identifier of symbool. Deze bestaat uit een letter, gevolgd door een of meer letters en/of cijfers. Voorbeelden hiervan zijn: I, ALPHA, P43, J4K5, HOUSEHOLDERTRIDIAGONALIZATION. Als uitzondering mag tevens de "single quote" (') gebruikt worden als "leesbaarheidskarakter". Dus: NOT'A'CHAR i.p.v. NOTACHAR. Een symbool mag bestaan uit 64 karakters, zodat
DONAUDAMPFSCHIFFFAHRTSMATROSENUNTERDECK en
DONAUDAMPFSCHIFFFAHRTSMATROSENAUFDECK nog onderscheidbaar zijn, wat bij andere systemen niet altijd het geval is.
We kunnen nu een assemblerprogramma leesbaar maken door i.p.v. de labels LL0, LL1, LL2 enz. suggestieve namen te gebruiken zoals ENTRY, PORT1, TIME2GO. En wat te denken van CMP@ERROR94 en JSR PRINT'1'SPACE.

Aangezien in het bronprogramma geen Basic-statements mogen voorkomen, kan dit programma ook niet geRUNd worden. In plaats daarvan is er een nieuw commando ASSEMBLE dat het programma in de huidige text space assembleert. Dit programma wordt met de normale editor aangemaakt en bevat voornamelijk datgene dat gewoonlijk tussen vierkante haken staat.
De eerste regel bevat meestal een commentaar met de programmanaam en in de tweede regel wordt bijvoorbeeld bepaald waar de vertaalde code terecht moet komen. Dit gebeurt met * = #XXXX, wat het equivalent is van DIM P(-1) of P = #XXXX.
Het begin kan er dus zo uitzien:

```
10 \ PROGRAMMA BLOCK MOVE
20 * = #3000 \ vertaal naar #3000
```

Hierna zullen i.h.a. enkele symbolen geïnitieerd worden:

```
30 .FROM  = #70
40 .UNTIL = FROM + 2
50 .TO    = UNTIL + 2
```

FROM krijgt de waarde #70, UNTIL de waarde #72 en TO de waarde #74. Zo hebben we de beschikking over drie pointers die we met FROM en FROM+1 enz. kunnen aanroepen.
We kunnen i.p.v. pointers ook adressen en getallen initiëren:

```
60 .WRCH = #FFF4
70 .BELL = 7
```

We kunnen nu in assembler programmeren zoals we dat gewend zijn, waarbij we deze symbolen kunnen gebruiken, b.v. aan het eind van het programma:

```
2000:SLUIT'AF LDA@BELL
2010          JSR WRCH
2020          RTS
```

Bij het initiëren van symbolen moet bedacht worden dat er enkele namen gereserveerd zijn voor andere doeleinden. Deze namen zijn: BYTE, ASCII, WORD, DBYTE, PRON, PROFF, SCRON, SCROFF, END en OPTION. Het is dus niet toegestaan om b.v. .BYTE = #AA te schrijven. Deze 10 zgn. directives geven aanwijzingen aan de assembler om iets speciaals te doen. Zo zorgt .PRON ervoor dat de printer aangezet wordt en .PROFF zet deze weer uit. Deze twee werken alleen tijdens de tweede pass. .SCRON en .SCROFF bedienen het scherm. Worden deze niet gebruikt dan is het scherm uit bij de eerste pass en aan bij de tweede pass. .END geeft aan waar het bronprogramma eindigt en .OPTION doet voorlopig nog niets.
.BYTE leest 1 of meer waarden in en plaatst deze achtereenvolgens in het geheugen op de plek tot waar de assembler heeft vertaald. .WORD leest een adres en plaatst dit in het geheugen, low byte eerst. .DBYTE doet hetzelfde, maar nu het high byte eerst.
.ASCII tenslotte leest een string in en plaatst de afzonderlijke karakters ervan in het geheugen. Deze string wordt voorafgegaan door en afgesloten met een zelf te kiezen karakter dat niet in de string mag voorkomen. Dus .ASCII/TEKST/ en .ASCII"TEKST" zijn gelijkwaardig. We kunnen hiermee b.v. de volgende constructie maken:

```
  .TEXT'END = #EA
  .PRNEXT   = #F7D1
 :WAIT'FOR'KEY'PRESS
   JSR PRNEXT
  .ASCII-PRESS ANY KEY TO CONTINUE-
  .BYTE LF,CR,TEXT'END
   JSR OSRDCH
   RTS
```

De symbolen LF (Line Feed), CR (Carriage Return) en OSRDCH (#FFE3) worden hier niet geinitiëerd, omdat deze standaard aanwezig zijn, evenals OSWRCH, ERRVEC, MASK, PORT, RDCVEC, WRCVEC.
Hoewel er nog veel meer faciliteiten aanwezig zijn, wil ik hier besluiten met de veranderde constructie voor het initiëren van pointers. De bovenstaande routine kan b.v. ook zo:

```
  .PTR = #70
 :WAIT'FOR'KEY'PRESS
   LDY@#FF      \ initiëer Y-reg.
   LDA@TEKST>   \ high byte van adres TEKST
   STA PTR+1    \ naar #71
   LDA@TEKST<   \ low byte van adres TEKST
   STA PTR      \ naar #70
 :CHR INY       \ verhoog index
   LDA (PTR),Y\ lees karakter
   JSR OSWRCH \ druk af
   CMP@CR       \ was het CR?
   BNE CHR      \ nee, volgende karakter
   RTS          \ ja, klaar
 :TEKST

 .ASCII/PRESS ANY KEY TO CONTINUE/
 .BYTE LF,CR
 .END
```

Ik dacht dat ik met deze beschrijving de belangrijkste zaken te pakken had. De overige faciliteiten zijn voornamelijk verfijningen van bestaande constructies. Ik zal daar wellicht een volgende keer over schrijven.

Hoe maak ik mijn programma leesbaar

Een tegenhanger van het artikel uit Acorn Nieuws '83-5 (lees het er maar eens naast: pag. 38/39).

Op dat artikel heb ik twee dingen tegen.
Ten eerste is het overbodig:
'BASIC' in zijn (haar?) pure vorm is sowieso ongeschikt om een optimaal leesbare listing te geven, de standaard Atom-BASIC doet daar nog een schepje bovenop: zeker als afkortingen worden gebruikt.
Ooit een echt goed leesbaar programma in A.N. gezien ?
Ten tweede is het ongewenst:
in een hobbyklub als de onze kan het nooit de bedoeling zijn om programma's te publiceren/verspreiden waarvan de werking met alle mogelijke middelen wordt verhuld.

Echter de toekomst lonkt.
Een BASIC programma kan redelijk leesbaar gemaakt worden gemaakt, Hoegel-BASIC geeft nog meer mogelijkheden daartoe, en geheugenruimte wordt steeds minder kostbaar.
En mensen die eenmaal een goed leesbaar programma hebben gezien eisen (terecht) dat alle programma's doorzichtig en begrijpelijk worden.
Om de bedreiging van on(be)grijpbare programma's het hoofd te bieden, moet de programmeur zich aan enkele strikte regels houden.

1. een programma is een opstel.
doe er alles aan om het idee de kop in te drukken, dat een voltooid programma niet net als een opstel een titel, een datum, de naam van de auteur , en een beginalinea over de bedoeling moet vertonen. Zonder zo'n introduktie weet de lezer niet wie 't programma schreef en waarom. Hij weet niet wat die variabelen allemaal voorstellen. De kans is groot dat hij niet de moeite zal nemen om dat allemaal uit te zoeken. Onprettig, want niemand zal je programma op z'n juiste waarde weten te schatten !

2. kommentaar is broodnodig.
REM betekent REMEMBER: zeker in Atom-BASIC met z'n onmogelijkheid om de variabelen een suggestieve naam mee te geven is het onontbeerlijk de lezer er aan te herinneren wat de diverse variabelen hier precies te betekenen hebben.

3. streel de ogen.
Normale schrijftaal heeft allerlei mogelijkheden om de tekst leesbaar te maken.
De programmeur moet waar zinvol spaties gebruiken. Zo worden die lussen makkelijk herkenbaar.
Bedenk dat extra spaties nauwelijks extra tijd kosten, het voluit schrijven van de instrukties daarentegen iets meer, maar snelheid is pas in het ALLERLAATSTE stadium van belang (en zeker niet op het moment van publikatie).

4. volg zoveel mogelijk de logische gang van zaken.
Je moet regelmatig je programma kontroleren. Een klein beetje extra werk kan geen kwaad.
Gebruik de control-struktuur waar het probleem om vraagt: mede dank zij de P-charm hebben we de beschikking over DO-UNTIL, WHILE-WEND,

CASE-CEND, FOR-NEXT, XIF-THEN-ELSE.
Bovendien zijn PROCEDURES en FUNCTIES binnen ons bereik gekomen: kies zinvolle namen !
Bedenk dat de GOTO niet meer echt nodig is.
De goede programmeur zal de meeste voldoening smaken wanneer hij zelfs na enige tijd zijn oude programma's nog begrijpt. Want als hij ZELF dat niet meer kan, dan zal niemand het kunnen !
Vanaf nu heeft niemand dus meer een excuus om een onleesbaar programma te produceren.

Rob van Dort

aan redactie, auteurs en lezers

kortom: leden van de A.C.C.,

zeker in het laatste nummer van Acorn Nieuws viel mij op dat het nivo van de inzendingen overwegend hoog is: op zich zeer verheugend, dit betekent dat onze Atom bloeit als nooit tevoren.
Er is echter een 'maar': hoe meer specialistisch of ingewikkeld het artikel, hoe kleiner het publiek.
Bovendien valt op dat de auteurs een vrij vast groepje vormen met maar weinig 'nieuwe aanvoer'.
Wellicht schrikt het grote aantal ingewikkelde publikaties lezers af om een gewoon, bijzonder, gezellig, interessant etc. programma, artikel of probleem in te zenden. JAMMER !!!
Naar mijn persoonlijke smaak ligt de balans nu verkeerd.
We zijn een hobbyklub met bijna 900 leden en ieder van hen moet aan zijn 'trekken' kunnen komen !

Overwin eventuele onnodige en ongewenste schroom en passiviteit en laat uw stem horen. Er leven ongetwijfeld ideeen en problemen waarmee anderen uit de voeten zouden kunnen !!

Rob van Dort

MEMORY EDITOR

Het hier beschreven programma geeft ons de mogelijkheid een hex cq. ascii-dump te editen, waarbij de overeenkomstige bytes in het geheugen ook veranderen.
U kunt met de cursor toetsen over de hex dump lopen doch alleen over die stukken die data representeren. Als U een bepaald byte wilt wijzigen tikt U gewoon de nieuwe waarde in en deze zal in beide dump's (hex zowel als ascii) veranderen. Het programma test of een bepaalde invoer toegestaan is. Met behulp van de commando's CTRL-N en CTRL-P kunt U de volgende of vorige 32 bytes op uw scherm krijgen waarbij het huidige blok eerst in de gewijzigde versie weer in het geheugen opgeslagen wordt. Met behulp van CTRL-S springt de cursor van ascii naar hex of omgekeerd en CTRL-R tenslotte 'restart' het programma. Beeindigen van het programma gebeurt met het antwoorden van '0' op de vraag START ADDRESS?.
Het programma is zoveel mogelijk uit subroutines opgebouwd die voor zichzelf spreken. Het machine code gedeelte is nodig om de cursor toetsen te decoderen en te veranderen in de commando's CTRL-I, -J, -K en -H.

```
 10 REM MEMORY EDITOR
 20 DIMTT(6);GOSUB m
 30 PRINT$12"    *** MEMORY EDITOR ***"';GOSUB a;P.'
 40 PRINT"START ADDRESS       "$8$8$8$8$8$8;INPUT S
 45 IF S=0;GOSUB w;P.''"THANK YOU!"';END
 50 M=0
 60 GOSUB h
 70 M=1;X=0;Y=0;P.$30$10$10;GOSUB v
 80 DO GOSUB i;U.0
 90;REM INPUT ROUTINE
100 REM CTRL-N: NEXT PAGE
110 REM CTRL-P: PREVIOUS PAGE
120 REM CTRL-R: RESTART PROGRAM
130 REM CTRL-S: HEX<->ASCII
140 LINK TT5;C=?#80
150 IF C=#12;P.$30$10;G.40
160 IF C=#0E;GOSUB s;S=S+32;GOSUB h;GOSUB b;RETURN
170 IF C=#10;GOSUB s;S=S-32;GOSUB h;GOSUB b;RETURN
180 IF C=#13;GOSUB c;RETURN
190 IF C=#0B AND Y>0;P.$C;Y=Y-1;GOSUB d;RETURN
200 IF C=#0B;RETURN
210 IF C=#0A AND Y<7;P.$C;Y=Y+1;GOSUB d;RETURN
220 IF C=#0A;RETURN
230 K=3-M;REM K=3 OF 2
240 IF C<>9;GOTO 320
250 X=X+1
260 IF X=4 AND Y<7;GOSUB v;Y=Y+1;GOSUB d;RETURN
270 IF X=4;X=X-1;RETURN
280 FOR I=1 TO K;P.$9;NEXT I
290 GOSUB d;RETURN
300 X=X+1;IF X=4;GOSUB v
310 RETURN
```

```
320 IF C<>8;GOTO 400
330 X=X-1
340 IF X<0 AND Y>0;GOSUB p;Y=Y-1;GOSUB d;RETURN
350 IF X<0;X=X+1;RETURN
360 FOR I=1 TO K;P.$8;NEXT I
370 GOSUB d;RETURN
380 X=X-1;IF X<0;GOSUB p
390 RETURN
400 IF M=0;GOTO 510
410 IF C<#20 OR C>#7E;RETURN
420 P.$C" "
430 Q=!#DE;?#E1=0
440 P.$13$9$9$9$9
450 FOR I=0 TO X;P.$9$9$9;NEXT I
460 P.&C
470 !#DE=Q;X=X+1
480 IF X>3 AND Y<7;Y=Y+1;GOSUB v;RETURN
490 IF X>3;P.$8$8;X=X-1
500 RETURN
510 K=0;GOSUB j
520 IF C=0;RETURN
530 GOSUB d;K=W;LINK TTS;C=?#80
540 GOSUB j
550 IF C=0;GOTO 530
560 K=K*16+W
570 Q=!#DE;?#E1=0;P.$13
580 FOR I=0 TO 18;P.$9;NEXT I
590 FOR I=0 TO X;P.$9$9;NEXT I
600 IF K>#1F AND K<#7F;P.$K;GOTO 620
610 P."."
620 !#DE=Q;X=X+1;IF X<4;P." ";RETURN
630 IF X=4 AND Y<7;GOSUB v;Y=Y+1;RETURN
640 X=X-1;P.$8$8
650 RETURN
660vREM POSITION CURSOR
670 P.';P=20;IF M=0;P=6
680 FOR I=0 TO P
690  PRINT $9
700 NEXT I
710 X=0
720 RETURN
730hREM PRINT HEXDUMP
740 REM BASIS ADRES=S
750 B=S
760 P.$30$10$10$10
770 FOR I=0 TO 7
780  @=6;P.&B" "
790  FOR J=0 TO 3
800   @=0
810   IF B?J<#10;P."0"
820   P.&B?J" "
830  NEXT J
840  PRINT"  "
850  FOR J=0 TO 3
860   IF B?J<#20 OR B?J>#7E;P.". ";GOTO 880
870   C=B?J;P.$C" "
880  NEXT J
```

```
 890  B=B+4;P.'
 900 NEXT I
 910 RETURN
 920mREM MACHINE CODE INPUT
 930 REM ROUTINE
 940 FOR I=0 TO 5;TTI=-1;N.
 950 FOR I=1 TO 2
 960 DIMP-1;P.$21
 970[
 980:TT0 JSR#FE71;BCS TT0
 985 CPY @5;BEQ TT1-1 \ IGNORE LOCK
 986 CPY @14;BEQ TT1-1 \IGNORE COPY
 990 CPY @6;BNE TT2
1000 LDA @#FF;BIT#B001;BMI TT1
1010 LDA @8;RTS
1020:TT1 LDA @9;RTS
1030:TT2 CPY @7;BNE TT4
1040 LDA @#FF;BIT#B001;BMI TT3
1050 LDA @10;RTS
1060:TT3 LDA @11;RTS
1070:TT4 PHP;CLD;STX#E4;STY#E5
1080 TYA;JMP#FEB2
1090:TT5 JSR TT0;STA#80;RTS
1100];P.$6;NEXT I
1110 RETURN
1120pREM CURSOR POSITIONERING
1130 REM NAAR VORIGE REGEL
1140 P.$13$11;P=26;IF M=0;P=15
1150 FOR I=0 TO P
1160  PRINT $9
1170 NEXT I
1180 X=3
1190 RETURN
1200dREM DELAY LOOP
1210 FOR I=0 TO 5;WAIT;NEXT I
1220 RETURN
1230cREM CHANGE MODE
1240 REM ASCII(-)HEX
1250 P.$30$10$10;M=M:1
1260 FOR I=0 TO Y;P.$10;NEXT I
1270 P=18;IF M=0;P=3
1280 FOR I=0 TO P;P.$9;NEXT I
1290 FOR I=0 TO X
1300  IF M=0;P.$9$9$9
1310  IF M=1;P.$9$9
1320 NEXT I
1330 RETURN
1340sREM STORE THE CHANGED
1350 REM BYTES IN MEMORY
1360 P=?#DE+256*?#DF+?#E0
1370 ?P=(?P):#80
1380 P=#8060;Q=7
1390 FOR I=0 TO 7
1400  FOR J=0 TO 3
1410   C=P?Q
1420   IF C<7;C=C+9;GOTO 1440
```

```
1430   C=C-#30
1440   K=C
1450   C=P?(Q+1)
1460   IF C<7;C=C+9;GOTO 1480
1470   C=C-#30
1480   K=K*16+C
1490   S?(I*4+J)=K
1500   Q=Q+3
1510  NEXT J
1520  P=P+#20;Q=7
1530 NEXT I
1540 RETURN
1550bREM CURSOR POSITIONING
1560 REM BEGIN SCREEN
1570 X=0;Y=0;P.$30$10$10
1580 GOSUB v
1590 RETURN
1600jREM ACCEPT HEX INPUT
1610 IF C>#2F AND C<#3A;P.$C;W=C-#30;RETURN
1620 IF C>#40 AND C<#47;P.$C;W=C-#41+10;RETURN
1630 C=0;RETURN
1640aREM MENU
1650 P.'"--------------------------------"
1660 F.I=0 TO 7;P.$10;NEXT I
1670 P."--------------------------------"
1680 P."    CTRL-N = NEXT BLOCK"'
1690 P."    CTRL-P = PREVIOUS BLOCK"'
1700 P."    CTRL-R = RESTART"'
1710 P."    CTRL-S = ASCII<-->HEX"$30
1720 RETURN
2000wREM CLEAR DISPLAY AREA
2010 FOR I=#8060 TO #815F STEP 4
2020  !I=#20202020
2030 NEXT I
2040 RETURN
```

CASSETTE CATALOG

Het volgende programma is bedoeld voor mensen die (net als ik) m.b.v. de printer een overzicht willen maken van de programma's op hun cassettes. Het programma geeft de file-naam, het startadres, eindadres, executieadres en het aantal blokken. Als het eerste blok (blok 0) gemist wordt, dan wordt het beginadres berekend, en dit wordt aangegeven door een sterretje voor het begin adres. Als het laatste blok gemist wordt, kan dit uiteraard niet berekend worden en worden ergeen aantal streepjes uitgeprint. (zie o.a regels 170 en 340)

```
 10 REM CATALOG 5
 20 DIM RR(14)
 30 FOR N=0 TO 14: RRN=300: NEXT
 40 P.$21
 50 GOS.p
 60 GOS.p
 70 P.$6
 80 END
 90pDIM P(-1)
100[
110:RR0 JSR#CD54
120 SEC: JSR#FC38
130 JSR#CD54
140 JSR#F7D1
150]
160 $P="  FILENAME        START END  EXEC BLOCKS"
170 P?0=2:P?1=15
180 P=P+(LEN P)+1
190[
200 NOP
210:RR8 LDA@0: STA#DC: STA#91
220 JSR#FB8E:BVC RR11: BNE RR8
230 LDY@0
240 JSR#FBC9
250 JSR#FBE2
260 LDY@0
270:RR14 LDA#D9: STA#92: BEQ RR1
280 TAY
290:RR6 DEC#D5
300 DEY: BNE RR6
310 LDA@1
320 STA#91
330 JMP RR1
340:RR11 LDA@3:JSR#FFE9: RTS
350:RR1 LDA#ED,Y
360 CMP@#D: BEQ RR2
370 JSR#FFE9
380 INY: JMP RR1
390:RR2 STY#90
```

```
400 LDA@16; SEC; SBC#90
410 TAY
420:RR3 JSR#F7FD
430 DEY; BNE RR3
440 LDA#91; BEQ RR9
450 LDA@8; JSR#FFE9
460 LDA@#2A; JSR#FFE9
470 LDA@0; STA#91
480:RR9 LDA#D5; JSR#F802
490 LDA#D4; JSR#F802
500 JSR#F7FD
510:RR5 LDA#D8; BPL RR7
520:RR10 LDA@0; STA#DC
530 JSR#FBBE; BVC RR11; BNE RR10
540 JSR#FBC9
550 JSR#FBE2
560 LDA#92; CMP#D9; BPL RR12
570 JMP RR5
580:RR7 LDA#D4
590 CLC; ADC#D8; STA#D4
600 LDA#D5; ADC@0
610 JSR#F802
620 LDA#D4; JSR#F802
630 JSR#F7FD
640 LDA#D7; JSR#F802
650 LDA#D6; JSR#F802
660 JSR#F7FD
670 LDA#D9
680 CLC; ADC@1; STA#D9
690 LDA#DA; ADC@0
700 JSR#F802
710 LDA#D9; JSR#F802
720 JSR#CD54
730 JMP RR8
740:RR12 LDY@14; LDA@#2D
750:RR13 JSR#FFF4
760 DEY; BNE RR13
770 JSR#CD54
780 JMP RR14
790]
800 RETURN
```

Run voorbeeld:

```
FILENAME        START END  EXEC BLOCKS
DISASS          *2900 39C6 C2B2 0011
KLOK             3AE0 3BFB 3AE0 0002
ERROR POINTER    8200 84E7 C2B2 0003
```

FORTH

Hier leest U het eerste deel van een serie verhalen over de programmeertaal Forth, met de bedoeling U een beetje warm te laten lopen voor het gebruik van Forth. Ik wil eigenlijk een serie verhaaltjes leveren die naast de in Acorn Nieuws verschenen (en nog steeds verschijnende) inleiding in Forth, maar nu iets meer op onze eigen Atom gericht en met wat korte praktische voorbeelden en uitbreidingen.

Ik begin in dit eerste deel met het beschrijven van een ombouw van de standaard Forth in de richting van de 16k-kaart. Wat is namelijk het geval: De memorymap is geheel en al gericht op de 12k-Atom.(#2800-#3C00 en #8200-#9800). Als we Forth laden en runnen dan bevindt de dictionary met de systeemwoorden zich in het lage geheugen (#2800-#3C00) en de ruimte, waar onze eigen definities komen, in het hoge geheugen (vanaf #8C00). In dit stuk zit dan ook nog de tapebuffer (1/2k) en de ruimte voor de User variabelen.
Als we nu onze eigen definities gaan maken dan laden we eerst de screen-editor met behulp van het commando: 6 LOAD <CR> en (als we met graphics willen werken) het graphics-pack met 18 LOAD <CR>.
De nadelen hiervan zijn:

a) Onze beschikbare ruimte voor definities wordt voor een groot gedeelte opgeslokt door de screen-editor en de graphics commando's.
b) Als we een koude start doen zijn we de editor etc. echter weer kwijt en moeten we hem weer opnieuw laden.
c) Elke keer als we Forth laden moeten we APART de editor en het graphics pack laden.
d) Bij het gebruik van graphic commando's hebben we zeer veel last van storingen omdat ons programma zelf ook in het graphics geheugen ligt.
e) De 16k-kaart wordt in het gheel niet gebruikt.

Wat gaan we hieraan doen?:

a) Het systeem dusdanig wijzigen dat na laden van Forth en na een koude start onze eigen definities vanaf #3C00 komen. De tape buffer verleggen we naar #7CD0, zodat deze samen met de ruimte voor de User variabelen precies tot #8000 loopt. We hebben dan nog ruim 16k voor eigen definities.
b) De editor (en voor wie veel met graphics werkt, ook het graphics-pack) wordt vast in Forth opgenomen. Als we nu dus *RUN"FORTH" doen hoeven we verder geen screens meer te laden, maar staat de editor meteen tot onze beschikking.
c) We voegen verder nog enkele handige woorden toe aan Forth die ook weer aan de "koude start" dictionary toegevoegd worden. Bijv. SCOS en FCOS om vanuit Forth makkelijk de verschillende snelheden van het cassette operating system in te kunnen stellen.

Met bovenstaande wijzigingen zijn alle 5 de nadelen verdwenen of in ieder geval bijna verdwenen.

Een bijkomend voordeel is dat we nu ook mode 4 graphics kunnen gaan gebruiken.

Ik zal nu stapsgewijs de voorgestelde veranderingen behandelen.

1)Verplaatsing van de dictionary.

Deze verandering staat in z'n geheel in de manual (Forth Theory and Practice) op pag.127. Uit eigen ervaring weet ik echter dat bij de eerste (en mischien ook wel bij latere) druk deze pagina ontbreekt, vandaar dat ik toch de hele wijziging hier zal bespreken.
Wat gaan we doen?:
-De boot-up tabel aanpassen.
-Een nieuwe "FORGET" definieren.
Dit laatste is nodig omdat in forget een test op de geldigheid van adressen gedaan wordt, waarbij wordt aangenomen dat geldige adressen negatief zijn. Dit komt omdat we ons normaliter in het gebied #8200-#9800 bevinden, de adressen hebben hier allemaal een meest signifikant bit gelijk aan 1 (dus worden ze als negatief geinterpreteerd).
We definieren nu een nieuwe FORGET en kopieren die over de oude definitie. Omdat de nieuwe definitie korter is dan de oude geeft dit verder geen problemen.
De Boot-up tabel bevat constanten die bij een koude start in de User variabelen worden gezet zoals bijvoorbeeld de dictionary pointer, "DP". Deze Boot-up tabel moeten we dus wijzigen zodat de door ons gewenste startwaarden gezet worden.
In onderstaande listing kunt U op de plaats waar XXXX staat de door U gekozen bovengrens invulle (bijv.#8000 of #6000). Dit in verband met eventuele systeemscftware die op de geheugen kaart kan staan.

Listing.

```
COLD                    ( Maak de dictionary leeg        )
HEX                     ( en schakel om naar hex         )

3C00 DUP DP !           ( Zet de dictionary pointer op   )
DUP 1E +ORIGIN !        ( 3C00 en verander DP en z'n     )
                        ( boot-up waarde                 )

DUP 1C +ORIGIN !        ( Doe hezelfde voor FENCE        )
FENCE !

XXXX 3C -               ( Start adres van het nieuwe     )
DUP 10 +ORIGIN !        ( User variabelen gebied         )
                        ( = topadres-#3C                 )

DUP ' LIMIT !           ( Verander de waarde van de      )
204 - ' FIRST !         ( constanten die het begin en    )
                        ( einde van de tape buffer       )
                        ( aangeven.                      )
```

```
:TEMP                        ( Definieer de nieuwe FORGET   )
  CURRENT @ CONTEXT @        ( en noem deze tijdelijk TEMP  )
  - 18 ?ERROR                ( CONTEXT=CURRENT ?            )
                             ( nee dan error message 2      )
  [COMPILE] '
  DUP FENCE @                ( ligt het te verwijderen      )
  [ 2822 , ]                 ( woord beneden FENCE of in de)
  MAX < 15 ?ERROR            ( nucleus? ja dan error 21     )
  DUP NFA DP !
  LFA @ CURRENT @ ! ;

( Nu gaan we FORGET vervangen door TEMP                     )
' TEMP HERE OVER
' FORGET SWAP CMOVE          ( Copieer TEMP over FORGET     )

FORGET TEMP
COLD
```

Zo dat was punt 1. Het beste kunt U de nieuwe Forth nu saven zodat latere fouten ons werk niet teniet doen.

NB: *SAVE"FORTH1" 240 400
    *SAVE"FORTH2" 2800 3C00

2) Het vast opnemen van de Editor.

Deze aanpassing is vrij eenvoudig en gaat op de volgende manier:

```
6 LOAD                       ( Laden van de editor screens )
DECIMAL                      ( Schakel om naar decimaal    )

 ( Aanpassingen in de boot-up tabel                        )
LATEST 12 +ORIGIN !          ( top NFA wijzigen            )
HERE 28 +ORIGIN !            ( FENCE wijzigen              )
HERE 30 +ORIGIN !            ( idem voor de Dictionary     )
                             ( pointer                     )
' EDITOR 6 +                 ( en voor VOC-LINK            )
32 +ORIGIN !
HERE FENCE !                 ( Wijzig de huidige waarde van)
                             ( FENCE                       )
```

Na deze wijzigingen zal de Editor ook bij een "koude" start aanwezig zijn.

We kunnen de gewijzigde versie van Forth nu weer saven. Daartoe moeten we eerst de top van Forth weten (dwz het hoogste adres dat door de Forth dictionary in beslag genomen wordt).
Dit doen we op de volgende manier:

```
HEX                          ( schakel om naar hex         )
DP @ U.                      ( print de waarde van DP      )
```

Deze waarde noteren we (zeg XXXX) en gaan nu met behulp van <break> uit Forth. Daarna saven we Forth:
*SAVE"FORTH1" 240 400
*SAVE"FORTH2" 2800 XXXX
Als we Forth weer willen laden geven we eerst het commando:
*LOAD"FORTH1" en daarna:
*RUN"FORTH2"

3)De uitbreidingen.

We gaan nu de woorden definieren die niet standaard bij Forth geleverd worden maar wel erg handig zijn om in de dictionary te hebben.

a)COS

```
( inschakelen van 300 Baud cassette interface

HEX

: COS
    F8EF 0206 !
    F96E 020C !
    FAE5 020E !
    C2AC DUP 210 ! 212 !
    FBEE 0214 !
    FC7C 0216 !
    FC38 0218 !
    C278 021A !
    CR.
." CASSETTE OPERATING SYSTEM 1"
    CR
;
```

b)FCOS

```
( 1200 Baud cassetteoperating system.                              )
( Gaat uit van de Josbox in voet 4 van de schakel-                 )
( kaart. Indien U geen schakelkaart hebt of als bij U de           )
( Josbox in een andere voet zit dient U de regel *** weg           )
( te laten of het nummer 4 te vervangen door uw voetnummer         )
( Als U een ander 1200 Baud systeem hebt dan dat van de            )
( Josbox dient U ook de vectoren aan te passen                     )
 : FCOS
    4 BFFF C!              ( ***                                   )
    F8EF 0206 !            ( COMVEC-command line interpreter       )
    F96E 020C !            ( LOAD-FILE vector                      )
    A5E7 020E !            ( SAVE-FILE vector                      )
    C2AC DUP 210 ! 212 !   ( goed zetten van RDR- en STRVEC        )
                           ( alleen nodig indien men een DOS       )
                           ( gebruikt. In het andere geval         )
                           ( gewoon deze regel weglaten            )
    A527 0214 !            ( GET BYTE-vector                       )
    A592 0216 !            ( PUT BYTE-vector                       )
    FC38 0218 !            ( FIND-vector                           )
    C278 021A !            ( SHUT-vector                           )
    CR
." CASSETTE OPERATING SYSTEM 2"
    CR
;
```

c) DOS

```
  ( Indien er geen disk-systeem beschikbaar is dit commando    )
  ( gewoon weglaten                                             )
  CREATE DOS                   ( Linkt naar E000 waar de vectoren
voor DOS gezet worden )
      8E86 ,                   ( STX #86;\ Bewaar het X register   )
      20 C, E000 ,             ( JSR #E000;\ Initialiseer de DOS   )
      8EA6 ,                   ( LDX #86;\ Herstel het X register  )
      4C C, 2842 ,             ( JMP #2842;\ en spring naar NEXT   )
  SMUDGE
```

d) VLIST

```
  ( Dit is een slimmere versie van de ingebouwde routine die   )
  ( de gedefinieerde woorden niet alleen list maar dit ook     )
  ( nog netjes in kolommen onder elkaar doet.                  )


  DECIMAL

  : VLIST
      0 OUT C! 31 C/L !
      CR CONTEXT @ @
      BEGIN
       DUP ID. OUT C@ 10
       / 1+ 10 * DUP C/L @ 10 - >
       IF DROP CR 0 OUT C! 0
       THEN OUT C@ - SPACES
       PFA LFA @ DUP 0= ?ESC OR
      UNTIL DROP CR
  ;
```

e) EXTEND

```
  ( Uitvoeren van EXTEND heeft tot gevolg dat nieuw gedefi-    )
  ( nieerde woorden COLD-start bestendig worden.               )

  : EXTEND
      LATEST 12 +ORIGIN !
      HERE 28 +ORIGIN !
      HERE 30 +ORIGIN !
      HERE FENCE !
  ;
```

f) MONITOR

```
  ( Deze uitbreiding staat helemaal in de manual: pag 79.       )
  ( Ze geeft ons de mogelijkheid om vanuit Forth monitor com-   )
  ( mando's te geven. Zoals bijv. MONITOR CAT om een inhouds-   )
  ( opgave van de band te krijgen.                              )
```

g) BASIC

```
  ( Terug naar het Basic systeem zonder een <BREAK> te hoeven )
  ( geven. Dit spreekt dacht ik voor zich.                    )
```

```
CREATE BASIC
     20 C, FFED ,           ( JSR #FFED;\ Genereer CR,LF        )
     4C C, C282 ,           ( JMP #C282;\ en spring naar basic  )
SMUDGE
```

Deze uitbreidingen kunnen ook weer vast aan onze nieuwe Forth worden toegevoegd en wel door intypen van EXTEND <CR>. Daarna
In dit dit deel van een serietje over de programmeertaal Forth wil ik het hebben over het Tape systeem en de daarbij behorende screens.
Hoe schrijven we normaliter een programma in Forth? We beginnen met het opstarten van de editor met het commando: EDITOR <CR>, waarna we alle woorden uit de EDITOR VOCABULARY kunnen gebruiken. Als we op de gebruikelijke manier het programma hebben ingevoerd (zie voor een handleiding: Forth Theory and Practice hoofdstuk 8) kunnen we dit gaan testen en wel als volgt:
We toetsen in ENTER <CR>, de nieuwe woorden worden nu vanuit de tape buffer in de dictionary gecompileerd waarna we deze nieuwe woorden kunnen testen door hun naam en eventueel benodigde getallen in te voeren gevolgd door <CR>. Als alles nog niet naar behoren werkt gaan we dit met de Editor verbeteren. Deze manier van werken voldoet best aardig, er kleven echter toch een aantal nadelen aan die ik hier even wil toelichten:

a) Als we een programma uit een tijdschrift overtypen dan is dit meestal geschreven voor een andere Forth computer. Daar deze meestal grotere screens hebben (bv:16 regels van 80 karakters) kunnen we deze echter niet precies overtikken maar moeten we gaan comprimeren. Verder vervalt in dit geval het grootste deel van de testfase. Een makkelijkere manier om deze programma's in te voeren zou een gewone teksteditor zijn zoals bijv. de Wordpack. Deze heeft namelijk veel krachtigere edit mogelijkheden dan de editor van Forth. Zodat het tussenvoegen van vergeten stukken of het verbeteren van typefouten veel makkelijker wordt.
b) Een lang programma beslaat al snel enkele screens die dan in naam verschillen doordat hun nummer steeds met een wordt opgehoogd. We krijgen na verloop van tijd dus een bandje vol file's die geen betekenisvolle naam hebben. (Wie weet na enkele weken nog dat de file "657" zijn kalender programma bevat?). Het zou dus beter zijn om als een programma eenmaal klaar is dit een zinvolle naam te geven en op deze manier weg te schrijven.
c) Een Forth screen beslaat bij de Atom altijd 512 bytes onafhankelijk van het aantal karakters dat dit screen bevat. Het niet gebruikte gedeelte wordt namelijk gevuld met spaties. Omdat verder de screens als afzonderlijke files naar de band gestuurd worden zullen we geen voordeel hebben van de Josbox FCOS met verkorte pauzes tussen de blokken.

Conclusie:Het moet compacter, duidelijker en sneller.

Om bovenstaande nadelen te verminderen heb ik een nieuwe laad-routine geschreven die lange files aankan met een zinvolle naam. Het enige probleem is nu nog om van de al bestaande Forth programma's een aaneengesloten file te maken waaruit ook de overtollige spaties verwijderd zijn.
Dit doen we als volgt: neem als voorbeeld de bij het systeem geleverde editor. Deze beslaat de screens 006 t/m 014.

Laadt de screens afzonderlijk in het lage geheugen dmv de volgende commando's:

```
*LOAD"006" 2800
*LOAD"007" 2A00
*LOAD"008" 2C00
*LOAD"009" 2E00
*LOAD"010" 3000
*LOAD"011" 3200
*LOAD"012" 3400
*LOAD"013" 3600
*LOAD"014" 3800
```

We hebben nu een lange Forth file gekregen (elk Forth screen is nl precies 512 bytes lang). Als einde file aanduiding plaatsen we nog even #04 achter de file dmv:

```
?#3A00=#04
```

We zouden nu de file weer kunnen saven door
*SAVE"EDITOR" 2800 3A01
in te typen. Hierbij is echter het nadeel van de overtollige ruimte (overbodige spaties) nog niet opgelost.
Iedereen die de Wordpack heeft kan deze spaties mbv dit programma verwijderen. Anderen zullen hiertoe een Basic of Machinetaal programma moeten schrijven.

We hebben nu de nieuwe file maar kunnen deze nog niet inladen, dat kan wel na invoeren van het onderstaande programma.

LISTING:

SCR #026

```
( NEW LOADER -1- )
BASE @ HEX
: SCRLOAD LOAD ;
: MLOAD TIB @ >R IN @ >R
    TIB ! 0 IN ! INTERPRET
    R> IN ! R> TIB !
;
CREATE OSLOAD
    8E86 , 80A2 ,
    20 C, FFE0 , 8EA6 ,
    4C C, 2842 ,
SMUDGE
-->
```

SCR #027

```
: MESO
    CR
    ." WARNING: PROBABLY ILLEGAL"
    CR ." FILE FORMAT." CR
    ." <ESC> TO ABORT" CR
    KEY 1B = IF ABORT
             THEN
;
-->
```

SCR #028

```
O VARIABLE PTR
: RSP 8200 PTR !
    5 0 DO
    1000 O DO LOOP
          LOOP
    BEGIN PTR @ C@
     OD = IF 20 PTR @ C!
          THEN PTR @ C@ 04 =
           IF 00 PTR @ C!
           ELSE PTR @ C@
            20 - 0<
            IF MESO
            THEN
           THEN
     PTR @ DUP 1+ PTR ! C@ 0=
    UNTIL
; -->
```

SCR #029

```
( NEW LOADER -3- )
: LOAD
    ." FILENAME:"
    QUERY OD WORD
    HERE COUNT -TRAILING
    2DUP + OD SWAP C!
    DROP 80 ! 8200 82 !
    FF 84 C! OSLOAD RSP
    8200 MLOAD CR
    ." COMPILATION COMPLETED"
    CR
; BASE ! ;S
```

Toelichting.

SCRLOAD:Herdefiniering van het woord LOAD uit de standaard dictionary. Als we nu een screen willen laden toetsen we dus: 7 SCRLOAD i.p.v. 7 LOAD in.

MLOAD :Compileer de op #8200 geladen file in de dictionary.

OSLOAD :Machine taal routine die een file naar het graphics geheugen laadt.

MESO :Er wordt in het woord RSP gekeken of de file ook karakters bevat die kleiner zijn dan #20. Als dit het geval is wordt MESO aangeroepen om te waarschuwen dat de geladen file waarschijnlijk geen Forth Source file is.

RSP :Dit woord vervangt alle <CR> in de file door een spatie omdat Forth nl allen spatie als delimiter kent. De regel "1000 O DO LOOP" is alleen van belang voor diskgebruikers. Anderen kunnen deze weglaten.

LOAD :Het eigenlijke commando. Eerst wordt om een Filenaam gevraagd waarna de boodschap PLAY TAPE volgt. Na laden van het programma zal dit automatisch gecompileerd worden, waarbij de gewone foutmeldingen gegeven worden.

NB: Bovenstaande uitbreidingen werken alleen als Uw Forth systeem volgens de in Broodje 3 beschreven methode is aangepast. (Zie ook de Vergipsingen pagina).

Met deze kaart is het mogelijk om 8 kleuren te genereren op een kleurentelevisie in 2 groepen van 4.
Groep 1 bestaat uit de kleuren rood,blauw,groen en geel,groep 2 bestaat uit de kleuren cyaan,helrood,oranje en felgeel.Snel overschakelen tussen deze groepen wordt bereikt door het 'poken' van een geheugenlokatie,en via programmeren is het ook mogelijk om andere kleuren te verkrijgen.
Verderop in deze handleiding vind U ook 4 voorbeeld programma's.
Verder wordt er bij deze handleiding een schema geleverd,een tekening over hoe de kaart op de ATOM wordt aangesloten,een komponenten lijst en enkele praktische tips.
Wanneer U de komponenten heeft aangeschaft kunt U ze op de print-plaat vastsolderen zoals daarop staat aangegeven.
De verbinding tussen de print en de ATOM kan het best en mooist worden gemaakt met een klein stukje flat-cable en een moeder-connector voor PL4.In figuur 2 en 3 staat aangegeven welke pin met welke plaats op de print wordt verbonden.
Wanneer U de kaart niet gebruikt en C2 verwijdert heeft uit de ATOM kunt U niet Uw TV aansluiting gebruiken.Als U echter de kaart laat zitten en er een zwart/wit TV op aansluit werkt deze als normaal.
Als de kaart aangesloten is en U heeft er een kleurentelevisie op aangesloten dan moet U deze op UHF zetten en hem daarna vrij kritisch afstellen (dit kan dus inhouden dat de kaart niet goed werkt op een televisie die zichzelf automatisch afstelt!!)Hierna kunt U aan de trimmer
draaien totdat U kleur krijgt.(een groen bord op een rode achtergrond.)Als U geen kleur verkrijgt loop dan de bijgeleverde eventuele foutenlijst af uit figuur 5.
Een kleine test die U nu kunt doen is:
Type in:CLEAR4:?#B000=#D0:F.N=#8000 TO #9800 S.4:!N=#FFAA5500:N.
Nu moet U vertikale strepen van rood,blauw,geel en groen zien.
Als U nu ingeeft ?#B002=8 ziet U de tweede groep kleuren verschijnen.(cyaan,oranje,felrood en helgeel.)
Nu werkt de kaart dus en de ACC-DELFT wenst U veel plezier.

FIG.5

Wat te doen als hij niet werkt?

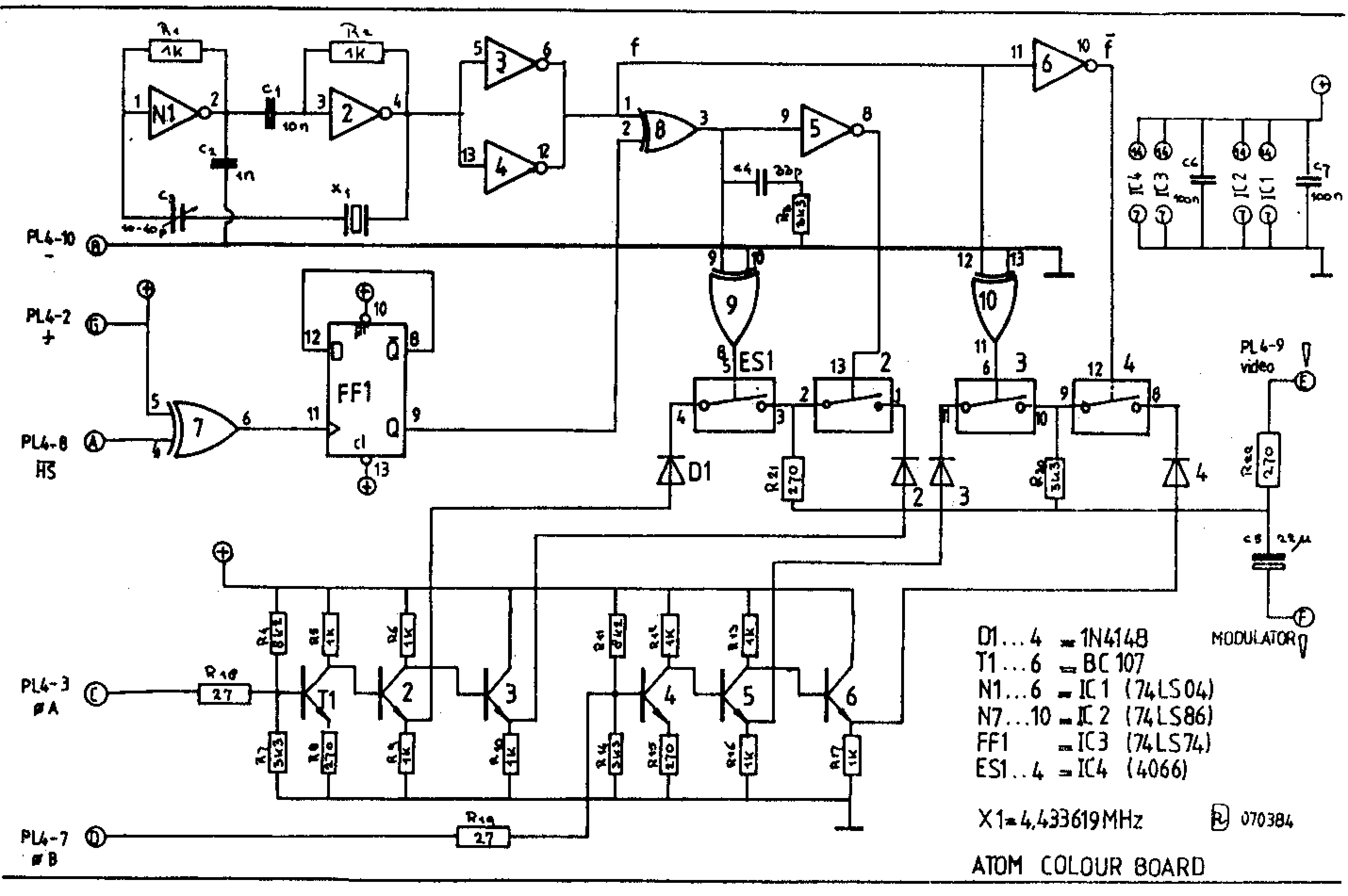
1.De kleurkaart trekt ongeveer 40 mA en daarom moet er een goede voeding gebruikt worden.Ook als de voeding onder de 4,8 volt staat zal de kaart niet werken!

2.Als U geheel geen beeld heeft controleer dan of de draad naar C2 wel helemaal goed zit.

3.Als U wel beeld maar geen kleur heeft controleer dan of de flat cabel goed is aangesloten, of de transistors goed zitten en of de trimmer goed vast zit.(de trimmer is een heel gevoelig onderdeel!)

4.Goed solderen is ook belangrijk laat eventueel alle soldeerpunten nog een keer vloeien.

5.Indien U,in verband met Uw printer,link 5 van de computer heeft onderbroken,zult U alleen de eerste groep kleuren kunnen krijgen,?#B002=8 zal dan geen effekt hebben.



PL4-10 −
PL4-2 +
PL4-8 HS
PL4-3 ØA
PL4-7 ØB
PL4-9 video
MODULATOR
N1
FF1
ES1
D1
T1
D1…4 = 1N4148
T1…6 = BC 107
N1…6 = IC1 (74LS04)
N7…10 = IC2 (74LS86)
FF1 = IC3 (74LS74)
ES1..4 = IC4 (4066)
X1 = 4,433619 MHz
070384
ATOM COLOUR BOARD

BIJ HET SCHEMA VAN DE KLEURENKAART

ONDERDELENLIJST:

weerstanden (1/4W)
R1,2,5,6,9,10,12,13,16,17 :1K
R3,7,14,20 :3K3
R4,11 :8K2
R8,15,21,22 :270 ohm
R18,19 : 27 ohm

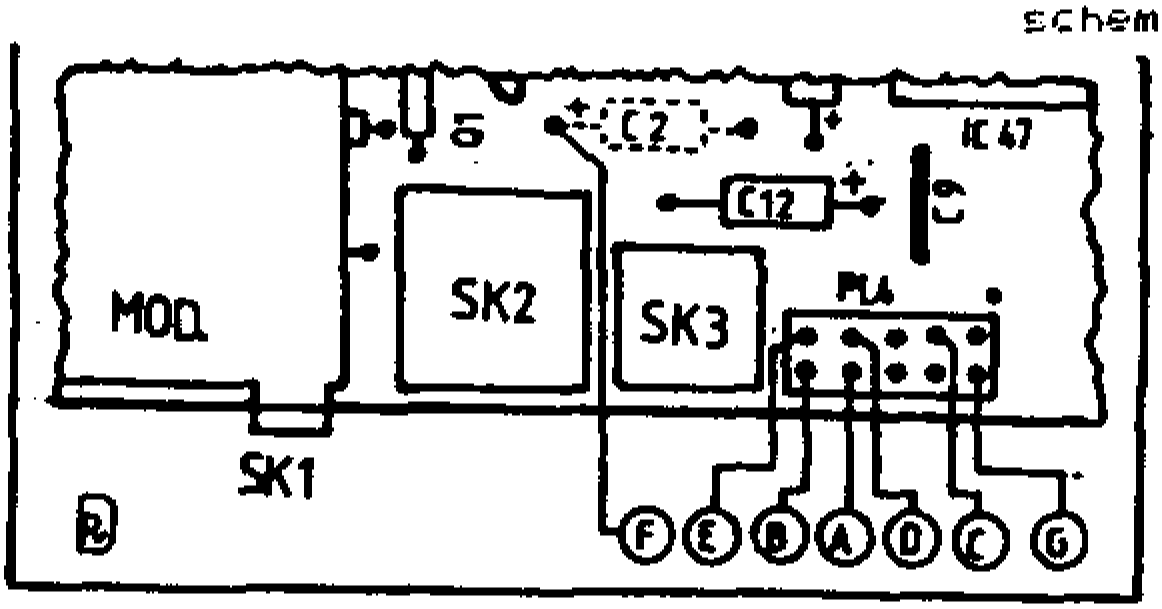
condensatoren
C1 : 10n
C2 : 1n
C3 : 10...40p trimmer
C4 : 33p
C5 : 22uF / 10V ELKO
C6,7 :100n

halfgeleiders
D1..4 :1N4148
T1..6 :BC107 (BC108,BC109,BC237, etc.)
IC1 :74LS04
IC2 :74LS86
IC3 :74LS74
IC4 :4066

diversen
X1 : 4.433619 MHz kleuren TV 'sub carrier' kristal

NB In de computer moet de condensator C2 (22uF) verwijderd worden. Punt F van de print wordt nu aangesloten op de punt waar eerst de + van C2 zat. (zie figuur)

ACC-DELFT.
schema: R.van Drunen.



KLEUREN-KAART

De kleurenkaart die de regio Delft heeft gemaakt, werkte bij mij niet zo best.
Het bleek echter mogelijk om met bescheiden veranderingen en geringe kosten (ong. f 5,-) mooie kleuren uit de kaart te krijgen. Om een kort verhaal lang te maken zal ik eerst de werking van het PAL-kleursysteem en de kleurenkaart bespreken en daarna de veranderingen in het schema.
Het Systeem.
Een mens kan slechts 3 kleuren waarnemen nl. rood, groen en blauw; de rest is slechts een mengsel van deze drie. Er zijn dus 3 elektrische signalen nodig voor kleurentelevisie.
Omdat zwart/wit kijkers er ook gebruik van maken moet 1 van de drie een z/w signaal zijn. De andere zijn dan verschilsignalen nl. rood-wit en blauw-wit.
Door optellen en aftrekken zijn dan de 3 kleuren afzonderlijk te krijgen.
Het beeld wordt in lijnen geschreven en het beeldsignaal geeft de variatie in de lichtsterkte weer tijdens het schrijven. De variatie kan heel snel zijn en daarom is de benodigde bandbreedte van het z/w signaal hoog nl. ong.0-4 MHz (max. 4 miljoen trillingen per sec.).
Voor kleur zouden we nog 2 signalen met elk een 4 MHz band nodig hebben maar dit zou teveel bandbreedte kosten en daardoor het aantal beschikbare kanalen voor de talrijke zenders te klein maken. Daarom moeten de kleursignalen het met een klein bandje stellen.
Dit betekent wel verlies aan snelheid wat zichtbaar wordt als onscherpte in de kleurovergangen.
Het PAL-systeem.
Als men 2 signalen met elkaar vermenigvuldigt ontstaan de som- en verschilfrekwenties, b.v. als men een beeldsignaal met een sinus(draaggolf) van b.v. 500 MHz vermenigvuldigt krijgt men een draaggolf met 2 zijbanden. zie fig.1. Dit is een AM-signaal in de uhf-band wat zo zou kunnen worden uitgezonden.
Deze techniek van het verplaatsen van een band gebruikt men ook om de kleursignalen bij het z/w signaal te kunnen voegen zonder dat ze elkaar storen.
De beide kleursignalen maken gebruik van dezelfde draaggolf van 4,43 MHz maar wel onderling 90 graden faseverschoven. Daardoor zijn ze in de tv weer uit elkaar te halen. De draaggolf wordt uit de kleursignalen verwijderd. Omdat hij in de tv nog nodig is wordt hij aan het begin van iedere beeldlijn even toegevoegd (de "burst"). Het volledige spectrum van het videosignaal ziet er uit als op fig.2. Om de kleursignalen nog beter te kunnen scheiden wordt nog een truukje uitgehaald: het rood-wit signaal wordt op een draaggolf gezet die bij ieder nieuwe beeldlijn omgedraaid wordt. In de TV worden een beeldlijn en zijn voorganger samen genomen: het verschil en de som levert zo de afzonderlijke kleursignalen op. Hiervoor is een vertragingslijn nodig die precies 1 beeldlijn vertraging geeft. De Kleurenkaart Fig.3 geeft het blokschema van de herziene kleurenkaart. De oscillator geeft de kleurdraaggolf, de 2-deler deelt de lijnfrekwentie t.b.v. de omkering van de draaggolf van het

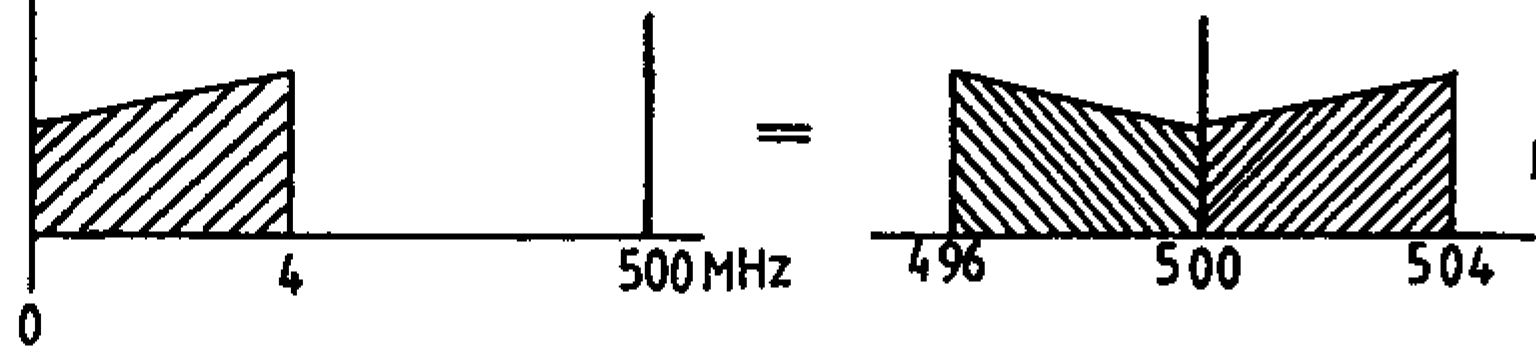


FIG1

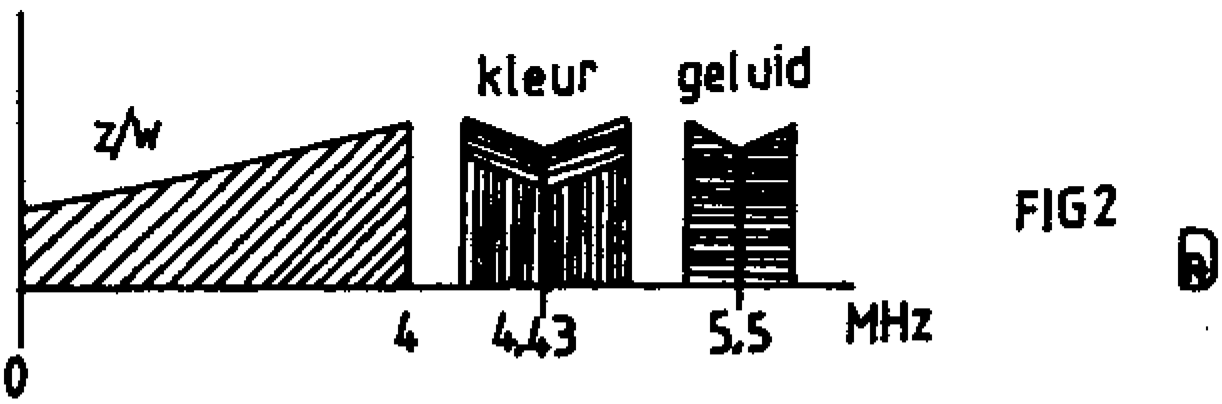


FIG2

rood-wit signaal. "X" is de vermenigvuldiger en in "-" wordt de draaggolf verwijderd. Daarna worden de kleursignalen gebufferd en bij het z/w signaal gevoegd. Fig.4 geeft het volledige schema. De plaats van de onderdelen komt ongeveer overeen met die in het blokschema. Over de voedingspootjes van IC 1 moet een condensator van ong.100 nF keramisch om parasitaire oscillatie te voorkomen. De c van 1 nF is niet nodig. C4 (33pF) is veel te klein om 90 graden vertraging te geven. R3 moet weg (kortsluiten). De CMOS IC4 is te traag of krijgt te weinig spanning; IC4 dus eruit. De kleurversterkers (T1-T6) werden overstuurd (T1,4) en hadden geen goed symmetrische instelling (T2,3/T5,6). De beide C's van 220 pF zijn een compromis. Neemt men hiervoor beide 270 pF dan zijn letters minder scherp maar ze hebben minder last van soms voorkomende storing. Eerst de trimmer in de oscillator afregelen, daarna de beide instelpotmeters en de trimmer over de instelpotmeter. Als de 1/8 w. weerstanden rechtop gezet worden passen ze in de gaatjes voor IC4. De scherpte in de kleurovergangen is nog niet optimaal. Dit komt doordat het kristal van de VDU (3,58 MHz) de Amerikaanse norm voor de beeldlijntijd oplevert; deze verschilt 0,8 % van de PAL-norm. Daardoor deugt de vertragingslijn in de TV niet. Met een 3,55 MHz kristal zou het beter zijn. Met een spoel is het 3,58 MHz kristal omlaag te verstemmen. Misschien is dit de moeite waard. Overigens heb ik de kleurenkaart in de ATOM ingebouwd.

Peter Ehrlich

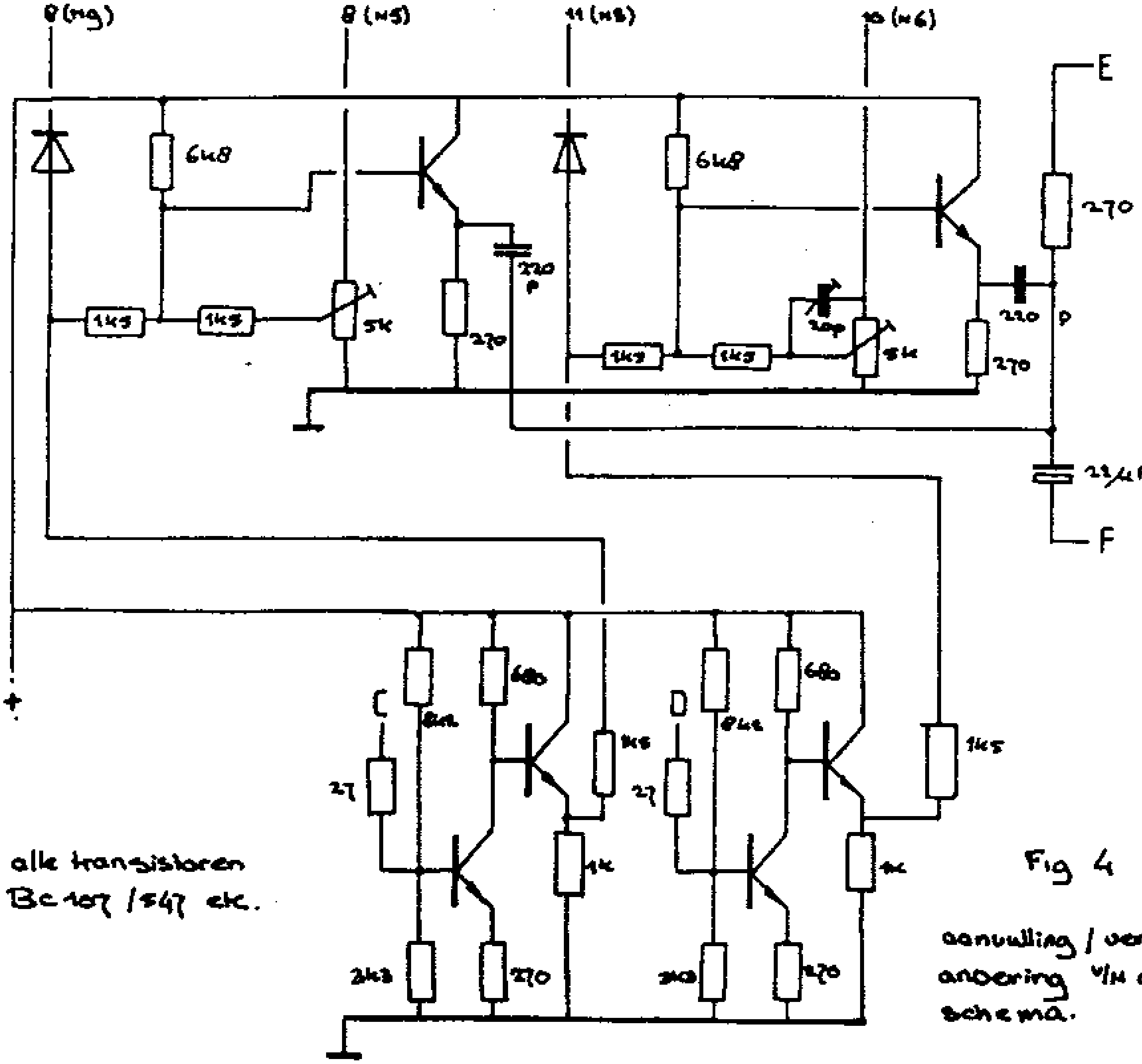


Fig 4

aanvulling / verandering v/h origineel schema.

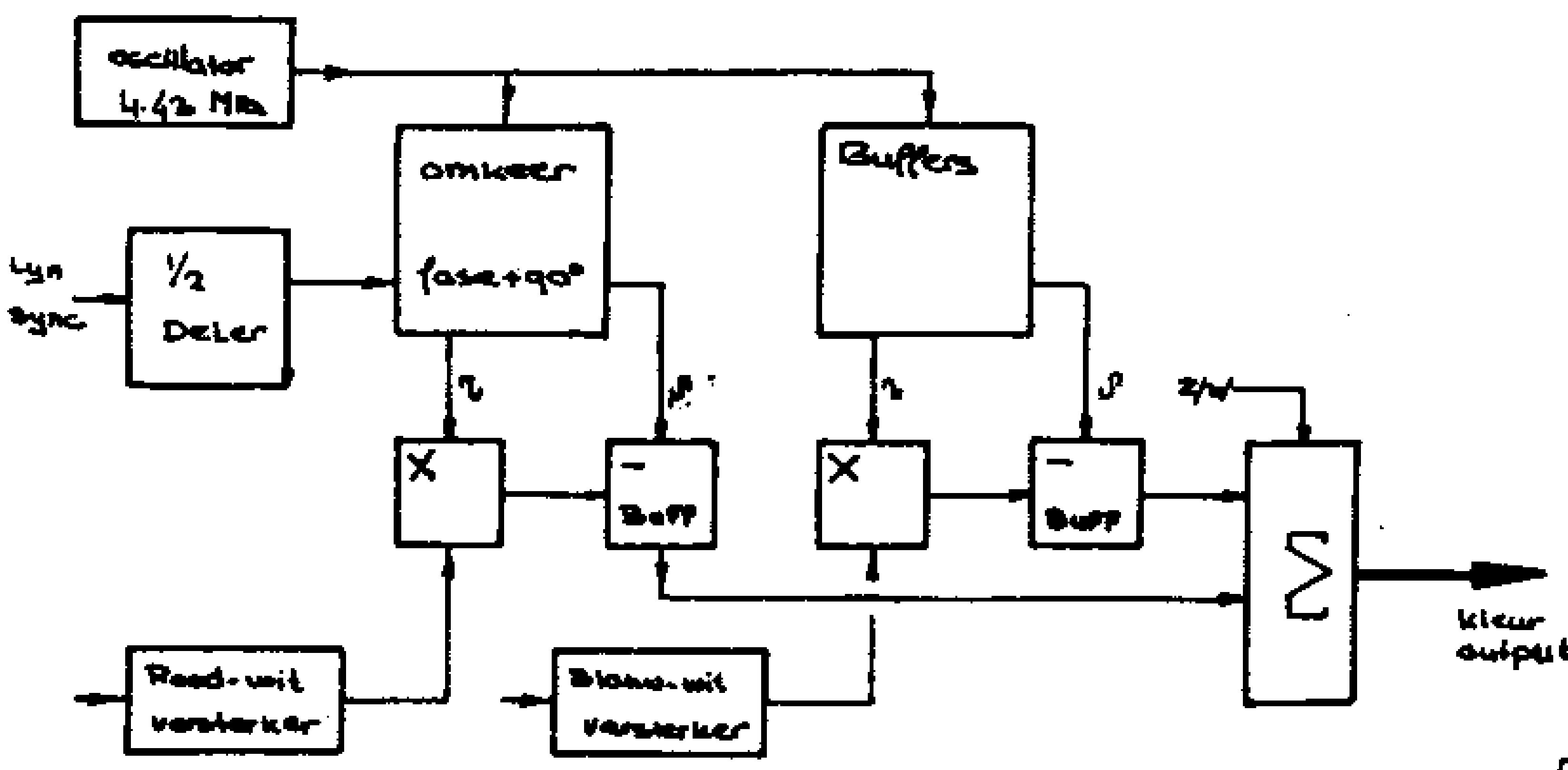


Fig 3 Blokschema.

140484

P - CHARME : EEN PAAR OPMERKINGEN

Als alles loopt zoals het lopen moet, dan is uw regio vanaf ongeveer 19 april reeds in het bezit van een verbeterde versie van de P-Charme interpretor. Mede door de grote complexiteit van dit 4k (!) programma zijn er stiekem enkele foutjes naar binnen geslopen. U kunt zelf controleren of u versie-nummer 171 bezit (de nieuwe versie) door het volgende programma te runnen:

```
10 PROGRAM TEST (P-CHARME MET OA. CORRECTE 1200 BAUD)
20
30 Z=171
40 FOR I=#A000 TO #AFFF
50   Z=Z:?I
60 NEXT I
70 XIF Z=CH"P-CHARME" THEN PRINT "OK"'
80 ELSE PRINT "ERROR IN P-CHARME"$7'" (SORRY!)"'
90 END
```

Ook in de handleiding zijn enkele kleine foutjes geslopen:

pag. 9 In het voorbeeld bij het INKEY statement staat ten onrechte ' UNTIL $A="." '. Dit moet uiteraard zijn ' UNTIL A=CH"." '(de beschrijving van het INKEY statement klopt echter wel).

pag 31 In de functie MID moet 'kleiner-dan' worden vervangen door 'kleiner of gelijk aan'. Zie de beschrijving van MID op pagina 53 van deze Acorn Nieuws.

pag 38 & 41 Hier mist een error-melding die bij het samenstellen van de lijst met error-nummers over het hoofd is gezien. Het betreft hier error 84: Illegal FUNCTION assignment. Het toekennen van een waarde aan een functie mag nl. alleen gebeuren op het niveau van de functie zelf (in de praktijk wil dat zoveel zeggen als 'functiewaarde toekenning moet gebeuren tussen bijbehorende FUNCTION en FEND').

Verder zijn er nog enkele misschien overbodige opmerkingen te maken, die blijkbaar niet allemaal voor de hand liggen:

1) TOP moet goed staan voordat een programma met behulp van de cursortoetsen of de EDITOR kan worden gewijzigd of met RENUM mag worden gerenumberd. Wees er op bedacht dat de EDITOR en RENUM traag kunnen worden als het programma een redelijke lengte heeft.
2) De ingebouwde functie, de zelf gedefinieerde functies en de procedures kunnen alleen binnen een programma worden gebruikt. Hieronder vallen ook de multi-dimensionale array's. Voor wat betreft de functies en array's is er echter een truc. Zie hiervoor pagina 50 ev. in dit nummer (8- statement).
3) Bij XIF hoort altijd een ELSE. de regel achter ELSE mag echter leeg zijn (of bv. worden opgevuld met een zinnig REM statement).

Vrijwel zonder uitzondering zijn de reacties op P-Charme zeer positief. Hiervoor dank.

CONTROLIST

Hierby een kort programma waarvan ik zelf al heel wat plezier heb gehad.
Het programma geeft een listing van programma's en wel regel na regel en bovendien geeft het de controlcode's aan, indien deze in een programma staan.
Ingebouwd zit een teller welke de lokatie van de cursor aangeeft.
Vooral bruikbaar voor hen die geen XDUMP of HDUMP hebben, alhoewel ik het zelf geregeld gebruik, omdat met byv XDUMP de regelnummers nogal lastig te vinden zyn.
Na laden en runnen geeft U het beginadres van het te kontroleren programma op, (byv #2900) dan drukt U op SHIFT voor start en voor elke volgende regel.
Hopelyk kan iemand er iets mee doen!

A.PETERS REGIO ARNHEM e/o

```
 0 REM /// CONTROLIST ///
 0 REM A.PETERS WEHL c "83
 0 REM GEEFT LISTING REGEL NA
 0 REM REGEL EN ZET CONTROL-
 0 REM CODES OP SCHERM.
10 P.$12;IN."STARTADRES "A
20 @=5;Q=#DE;R=#1B8000;C=1
30 P."START:SHIFT"'
40wIF?#B001=127 G.p
50 G.w
60pB=?A;Z=!Q;!Q=R;P.&A;!Q=Z;A=A+1
70lIFB>30 IFC=1 P.$B;G.p
80 IFC=0 P.'B*256+?A;A=A+1;C=1;G.p
90 IFB=13 C=0;G.w
95 B=B+64;P." CTRL-";G.l
```

WORDPACK-FILE OP 4000

HET VOLGENDE PROGRAMMA VERHUIST DE TEKSTRUIMTE VAN DE "WORDPACK ROM" NAAR #4000. U KUNT DE KOMPUTER NU RUSTIG UITSCHAKELEN OF EEN "BREAK" GEVEN, ZONDER DAT DE INGETIKTE TEKST VERLOREN GAAT OF BESCHADIGD WORDT.
TER ONDERSCHEIDING MOET U NU AANROEPEN MET: ED40 IPV EDIT.

```
 0 REM EDIT>>>ED40 // A.PETERS
10 P.$12;F.A=#A000 TO #AB10
20 IF ?A=40;?A=64
30 NEXT
40 ?#0A7D=52;?#0A7E=49
50 P."ED40 KLAAR VOOR GEBRUIK"''
60 P."AANROEPEN MET: ED40  RETURN"
70 END
```

A.PETERS REGIO ARNHEM E/O

5 - SCHERM

Het onderstaande programma is overgenomen uit de "rundbrief 3/4" van de Duitse ACORN Computer CLUB.

Dit programma maakt het fout-zoeken in langere programma's een stuk gemakkelijker. De inhoud van het beeldscherm-geheugen wordt met de inhoud van 4 pseudo-geheugens uitgewisseld. Dit gebeurt door de volgende toetsen in te drukken: CTRL-A of CTRL-D of CTRL-E of CTRL-P. Door het indrukken van een van deze combinaties wordt de inhoud van het beeldscherm-geheugen en het pseudo-geheugen uitgewisseld, door het opnieuw indrukken van dezelfde combinatie komt het systeem in zijn orginele toestand terug. Na een BREAK kan de routine opnieuw aangeroepen worden met LINK#2890.

```
 10REM VIJFVOUDIG SCHERM   DOOR KARL-HEINZ RATTAY
 20REM                     7800 FREIBURG, 02.08.82
 60DIM LL4
 80F.Z=1 TO 2
 90P=#2890
100[
110 LDA@#28;STA#20B
120 LDA@#A0;STA#20A
130 LDA@0;STA#90;STA#92
150 JSR#FE94;TAX;LDA@#82;CLC
160 CPX@1;BEQ LL3;CPX@4;BEQXLL2;CPX@5;BEQ LL1;CPX@16;BEQ LL0
180 TXA;RTS
200:LL0 ADC@2
210:LL1 ADC@2
220:LL2 ADC@2
230:LL3
240 CLC;STA#91;ADC@1;STA#93;LDY@0
260:LL4 LDX#8000,Y;LDA(#90),Y;STA#8000,Y;TXA;STA(#90),Y
270 LDX#8100,Y;LDA(#92),Y;STA#8100,Y;TXA;STA(#92),Y
280 INY;CPY@0;BNE LL4
290 LDA@0;RTS
400]
410 N.;P.$6;LINK#2890;E.
```

MEMORIE

```
 10 DIM PP48,MM2;H=#3B00
 20 Z=0;O=0
 30 FOR N=1TO48;PPN=0;NEXT
 40 FOR L=65 TO 88
 50 GOS.r;GOS.r
 60 NEXT;CLEAR0;GOS.m
 70 GRMOD;?#E1=0
 80 P.$30"        **** MEMORISE ****"''
 90 P."           1 2 3 4 5 6 7 8"'
100 FOR R=1TO6;P.'R
110 FOR K=1TO8;P." ",$128;NEXT
120 P.';NEXT
```




```
130 ?#2800=28;!#2801=#55555555
140 !#2805=#77775555;!#2809=#77777777
150 !#280D=#44447777;!#2811=#44444444
160 !#2815=#66666666;!#2819=#66666666
170 FOR X=69 TO 181 S.16
180 FOR Y=25 TO 145 S.24
190 MOVE X,Y;SHAPE#2800
200 NEXT Y
210 NEXT X
220 ON ERR GOS.740;P.$30'
230 DO Z=Z+1
240 FOR B=1TO2;P.$30';PLAY C"4,G"4
250 GOS.740;P.$30'
260 IN."RIJ,KOLOM:"A;R=A/10;K=A%10
270 IF R)6 OR K)8 P.$7$7$7;GOS.740;G.260
275 IF R(1 OR K(1 P.$7$7$7;GOS.740;G.260
280 IF (PP((R-1)*8+K))=0 P.$7$7$7;GOS.740;G.260
290 GOS.740;MMB=A
300 P.$10;F.Y=1TOR;P.$10$10;N.
310 F.M=1TO7;P.$9;N.
320 F.X=1TOK;P.$9$9;N.;P.$(PP((R-1)*8+K))
330 NEXT;W=300;GOS.w;P.$30
340 GOS.a
350 FOR B=1TO2;R=MMB/10;K=MMB%10
360 P.$10$10;F.Y=1TOR;P.$10$10;N.
370 FOR M=1TO7;P.$9;N.
380 FOR X=1TOK;P.$9$9;N.;P.$H,$30
390 NEXT;U.O=24
400 TXMOD;P.$12''"HET IS U GELUKT IN ............."''
405 W=100;GOS.w
410 P."           "Z" BEURTEN !!!!!"''
420 IN."NOG EEN SPEL "$H;IF $H="JA" THEN RUN
430 END
450rP=A.R.%48+1
460 IF PPP()0 GOTO r
470 PPP=L;RETURN
480wFORQ=1TOW;WAIT;N.;R.
490aC=MM1/10;D=MM1%10;E=MM2/10;F=MM2%10
500 IF PP((C-1)*8+D)()PP((E-1)*8+F) THEN ?H=128;?(H+1)=13;R.
510 $H=" ";O=O+1
520 PP((C-1)*8+D)=0
530 PP((E-1)*8+F)=0
540 RETURN
550mP.$12"               MEMORIE"'';?#E1=0
560 P."EEN SPEL WAARBIJ UW GEHEUGEN"'
570 P."OP DE PROEF WORDT GESTELD."''
580 P."OP HET SPEELVELD KOMEN 24"'
590 P."LETTER-PAREN VOOR."'
600 P."AAN U DE TAAK OM IN ZO WEINIG"'
610 P."MOGELIJK BEURTEN ALLE PAREN"'"OP TE SPOREN."'
620 P.''''"                F.M."'
630 W=800;GOS.w;P.$30;CLEAR0
640 P."          spelregels"''
650 P."U KIEST EEN RIJ EN EEN KOLOM."'
660 P."HET DAARBIJ BEHORENDE KAARTJE"'
670 P."WORDT OMGEDRAAID EN BLIJFT"'
680 P."ZICHTBAAR."'"DAARNA KIEST U EEN TWEEDE KAART."
690 P."BEIDE KAARTJES BLIJVEN EEN"'
```

```
700 P."POOSJE ZICHTBAAR EN KEREN DAN"'
710 P."WEER OM."'"ZIJN DE TWEE LETTERS ECHTER"'
720 P."GELIJK DAN VERDWIJNEN ZE."'
730 LINK#FFE3;RETURN
740 P.$30'"                         "'$11;RETURN
```

SOUND

```
 10 PROGRAM SOUND
 20
 30 ?#B002=2;REM 1MHZ
 40 E=1
 50 FOR I=1 TO 15
 60   FOR J=9 TO 60 STEP 4
 70     BEEP J,E;BEEP 69-J,E
 80   NEXT
 90 NEXT
100
110
120 FOR I=1 TO 5
130   FOR J=5 TO 30
140     BEEP J,2;BEEP 80-J*2,3
150   NEXT
160 NEXT
170
180 FOR I=1 TO 60
190   FOR J=0 TO 9
200     BEEP I+J,E
210   NEXT
220 NEXT
230 FOR I=59 TO 1 STEP -1
240   FOR J=0 TO 9
250     BEEP I+J,E
260   NEXT
270 NEXT
280
290 FOR I=1 TO 5
300   FOR J=4 TO 100
310     BEEP J,E
320   NEXT
330   FOR J=95 TO 9 STEP -1
340     BEEP J,E
350   NEXT
360 NEXT
370
380 FOR I=5 TO 1 STEP -1
390   K=1
400   FOR J=1 TO 63
410     BEEP J,2;BEEP K,E
420     K=(K+3)&63
430   NEXT
440 N .$L
460 FOR I=1 TO 3
470   FOR J=120 TO 2 STEP -2
480     BEEP J,E;BEEP J,2
490     BEEP J,3;BEEP J,4
500   NEXT
510 NEXT
```

```
 520
 530 FOR I=1 TO 3
 540   FOR J=20 TO 3 STEP -1
 550     FOR K=1 TO 3
 560       BEEP J,E;BEEP J+20,E
 570       BEEP J+90,E
 580     NEXT
 590   NEXT
 600 NEXT
 610
 620 FOR I=1 TO 5
 630   FOR J=45 TO 1 STEP -1
 640     BEEP J,E;BEEP J+10,E
 650     BEEP J+20,E
 660   NEXT
 670   FOR J=7 TO 40
 680     FOR K=1 TO 45 STEP J
 690       BEEP K,E
 700     NEXT
 710     FOR K=46 TO 1 STEP -J
 720       BEEP K,E
 730     NEXT
 740   NEXT
 750 NEXT
 760
 770 FOR I=1 TO 7
 780   FOR J=201 TO 255
 790     BEEP J,2
 800   NEXT
 810   FOR J=254 TO 200 STEP -1
 820     BEEP J,2
 830   NEXT
 840 NEXT
 850 FOR I=199 TO 1 STEP -1
 860   BEEP I,2
 870 NEXT
 880
 890 FOR I=1 TO 20
 900   BEEP 80+R.%4,5
 910   BEEP 0,A.R.%20+40
 920   BEEP 40+R.%3,3
 930   BEEP 0,A.R.%30+90
 940   BEEP 47+R.%3,5
 950   BEEP 0,A.R.%20+45
 960   BEEP 87+R.%4,3
 970   BEEP 0,A.R.%30+90
 980 NEXT
 990
1000 FOR I=1 TO 20
1010   FOR J=9 TO 60 STEP 4
1020     BEEP J,E;BEEP 69-J,E
1030   NEXT
1040 NEXT
```

```
1050
1060 FOR I=1 TO 10 STEP 2
1070   FOR J=10 TO 45
1080     FOR K=1 TO 100 STEP J/2
1090       BEEP K,E
1100     NEXT
1110     FOR K=46 TO J STEP -I
1120       BEEP K,E
1130     NEXT
1140   NEXT
1150 NEXT
1160
1170 END
```

DOOL HOF

```
 10 REM OVERNAME VRIJ MET
 20 REM BRONVERMELDING
 30 P.$12"   ***CREATIVE PROGRAMMING***"'
 40 P."             ***amazing***"'
 50 P."            ***labyrinth***"''
 60 P."MAXIMUM DIMENSIONS"'
 70 P."SCREEN:10* 7"'
 80 P."PAPER:40*60"'''''
 90aP.$11$11$11$11
100 IN."WIDTH"H
110 IN."LENGTH"V
120 IF H<2 OR V<2 P."meaninglessdimension"'';G.a
130 DIM A6,B32
140 $B="dimensiontoolarge"
150 IN."RESULT ALSO ON PAPER"$A
160 IF H>40 OR V>60 P.$B';G.a
170 IF ?A<>CH"Y" IF H>10 OR V>7 P.$B';G.a
180 ?#E1=0;P."                           "'
190 DIM WW(H*V),VV(H*V),MM3
200 F.I=1 TO (H*V*2);WWI=0;N.
210 X=1+A.R.%H;C=1;WWX=C
220 C=C+1;T=X;S=1;G.f
230bIF T<>H G.d
240 IF S<>V G.c
250 T=1;S=T;G.e
260cT=1;S=S+1;G.e
270dT=T+1
280eIF WW(S*H-H+T)=0 G.b
290fF.I=0 TO 3;MMI=1;N.
300 M=S*H-H+T;D=M+H;B=M-H;L=M-1;R=M+1
310 IF S=1 MM1=0;G.g
320 IF WWB<>0 MM1=0
330gIF S=V MM3=0;G.h
340 IF WWD<>0 MM3=0
350hIF T=1 MM0=0;G.i
360 IF WWL<>0 MM0=0
370iIF T=H MM2=0;G.j
380 IF WWR<>0 MM2=0
390jIFMM0=0 IF MM1=0 IF MM2=0 IF MM3=0 G.b
```

```
400 DO Y=A.R.%4
410 UNTIL MMY()0
420 G.(430+Y*10)
430 WWL=C;VVL=2;T=T-1;G.m
440 WWB=C;VVB=1;S=S-1;G.m
450 WWR=C;VVM=(VVM()0)+2;T=T+1;G.m
460 WWO=C;VVM=(VVM()0)*2+1;S=S+1
470mC=C+1;IF C()H*V+1 G.f
480 IF ?A=CH"Y" P.$2$30$27$54 IF H)20 P.$29$27$56
490 P.$12;F.I= 1 TO H
500 IF I=X P.".  ";G.n
510 P.".--"
520nN.;P.".“'
530 Z=1+A.R.%(H-1);VV(H*V)=0
540 VV(H*V-H+Z)=1+VV(H*V-H+Z)/2*2
550 F.J=1 TO V;P."I"
560 F.I=1 TO H
570 IF VV(J*H-H+I)(2 P."  I";G.o
580 P."   "
590oN.I;P.'
600 F.I=1 TO H
610 IF VV(J*H-H+I)%2 P.":  ";G.p
620 P.":--"
630pN.I;P.":"';N.J;P.$3;E.
  0 \\ R. OVEREEM //
```

```
 0 REM:PROGRAMMA OM BEGINADRES
 0 REM:VAN EEN REGELNUMMER TE
 0 REM:ZOEKEN EN DAAR BYV.
 0 REM:XDUMP TE STARTEN.
 0 REM:GEEF STARTADRES VAN
 0 REM:HET PROGRAMMA IN #....
 0 REM:DAN HET REGELNUMMER.
 0 REM:HET ANTWOORD KAN EVEN
 0 REM:OP ZICH LATEN WACHTEN.
 0 REM:c"02-84 A.PETERS WEHL
10P.''';@=0
20 IN."BEGINADRES "A
30 IN.'"REGELNUMMER "B
40bIF ?A=13 GOS.a
50 A=A+1;G.b
60aA=A+1;IF ?A=#FF P.'"REGEL "B" KOMT NIET VOOR!"''';G.10
70 Z=?A*256;A=A+1;Z=Z+?A
80 IF Z=B P.'"REGEL "Z" BEGINT OP #"&A-2;@=8;P.'';E.
90 R.
```